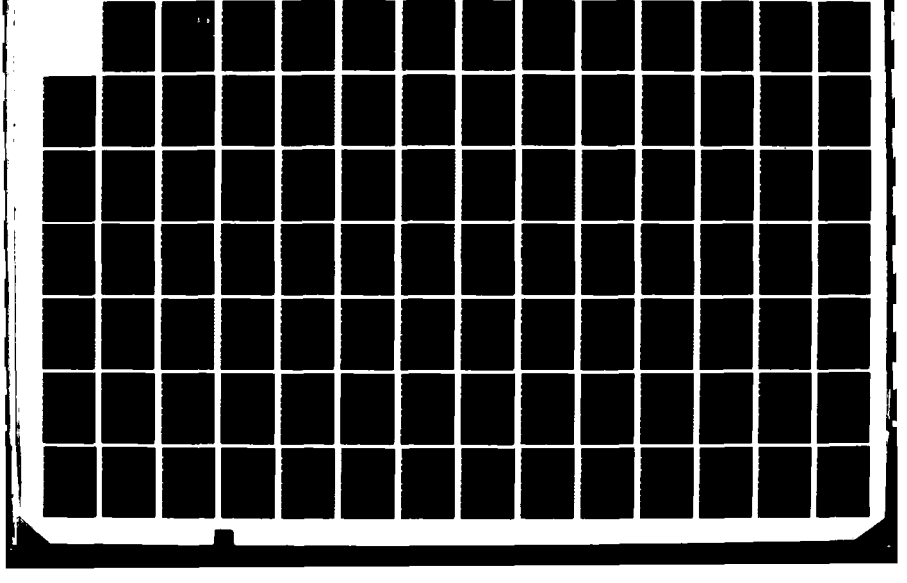
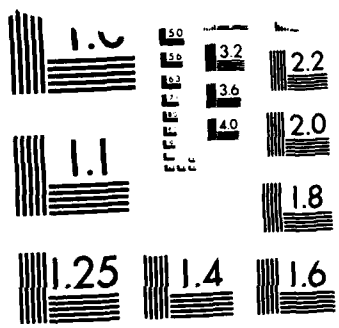
AD-A185 747 CONJUNCTIVE CONCEPTUAL CLUSTERING: A METHODOLOGY AND EXPERIMENTATION(U) ILLINOIS UNIV AT URBANA COORDINATED SCIENCE LAB R E STEPP SEP 87 UILU-ENG-87-2253 1/3

UNCLASSIFIED N00014-82-K-0186 F/G 12/9 NL





September 1987 UILU-ENG-87-2253

COORDINATED SCIENCE LABORATORY
College of Engineering

CONJUNCTIVE CONCEPTUAL CLUSTERING: A METHODOLOGY AND EXPERIMENTATION

Robert Earl Stepp, III

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-87-2253	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research & NSF
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Ave. Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy St. Arlington, VA 22217 NSF: 1800 G St., N.W. Washington, D.C. 20550
8a. NAME OF FUNDING/SPONSORING ORGANIZATION NSF and Office of Naval Research 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-82-K-0186 / NSF: MSC-82-05166
8c. ADDRESS (City, State, and ZIP Code) 800 N. Quincy St. Arlington, VA 22217 1800 G St., N.W. Washington, D.C. 20550	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION

11. TITLE (Include Security Classification)

Conjunctive Conceptual Clustering: A Methodology and Experimentation

12. PERSONAL AUTHOR(S)

Stepp, Robert Earl

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____		201

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	machine learning methodology, conjunctive conceptual clustering, soybean disease, blocks-world structures

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis describes a machine learning methodology called conjuctive conceptual clustering. The methodology can find conceptual patterns in data as illustrated by three sample problems. In one problem, the method is used to rediscover categories of soybean disease when given a collection of 47 descriptions of diseased soybeans having one of four diseases. In a second problem, the method is used to find categories underlying a collection of blocks-world structures. In a third problem, categories of objects having a more complex structure are determined and contrasted with categories generated by people.

The description method of conjunctive conceptual clustering forms clusters of objects (or situations) not on the basis of a numerical similarity measure but on the basis of the "conceptual cohesiveness" of one object to another. The conceptual cohesiveness between two objects depends on the descriptions of the two objects as well as the descriptions of other nearby objects in the given collection and concepts

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted.
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

which are available to describe object groups or object configurations as a whole. From a collection of objects, some background domain knowledge, and a goal or purpose for clustering, conceptual clustering generates a hierarchical classification composed of clusters of objects and corresponding conjunctive-form cluster descriptions (concepts). Conceptual clustering is one paradigm of "learning from observation" in which no teacher guides the learning process.

CONJUNCTIVE CONCEPTUAL CLUSTERING: A METHODOLOGY AND EXPERIMENTATION

Robert Earl Stepp, III, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1984

This thesis describes a machine learning methodology called conjunctive conceptual clustering. The methodology can find conceptual patterns in data as illustrated by three sample problems. In one problem, the method is used to rediscover categories of soybean disease when given a collection of 47 descriptions of diseased soybeans having one of four diseases. In a second problem, the method is used to find categories underlying a collection of blocks-world structures. In a third problem, categories of objects having a more complex structure are determined and contrasted with categories generated by people.

The described method of conjunctive conceptual clustering forms clusters of objects (or situations) not on the basis of a numerical similarity measure but on the basis of the "conceptual cohesiveness" of one object to another. The conceptual cohesiveness between two objects depends on the descriptions of the two objects as well as the descriptions of other nearby objects in the given collection and concepts which are available to describe object groups or object configurations as a whole. From a collection of objects, some background domain knowledge, and a goal or purpose for clustering, conceptual clustering generates a hierarchical classification composed of clusters of objects and corresponding conjunctive-form cluster descriptions (concepts). Conceptual clustering is one paradigm of "learning from observation" in which no teacher guides the learning process.

CONJUNCTIVE CONCEPTUAL CLUSTERNG: A METHODOLOGY AND EXPERIMENTATION

BY

ROBERT EARL STEPP, III

A.B., University of Nebraska, 1970
M.S., University of Nebraska, 1971

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1984

Urbana, Illinois

ACKNOWLEDGEMENTS

I wish to thank my advisor, Ryszard S. Michalski, for his help and encouragement throughout my studies at the University of Illinois. The opportunity to work on aspects of this research in a collaborative manner with him has been of particular benefit. This study owes much to theoretical foundations of inductive inference developed by him over the past years and to the many avenues of research he has directed. It has also been helpful to be affiliated with the Intelligent Systems Group and the Artificial Intelligence Laboratory in the University of Illinois Department of Computer Science.

I wish to thank my advisor and A.B. Baskin for their help reading the manuscript and their comments and criticisms. I would also like to think my friend Paul W. Baim for proofreading the manuscript, finding phrases of fuzzy rhetoric, and regularly calling long distance to ask "Why isn't it done yet?"

Special thanks go to Gayanne Carpenter for the pleasant way she has of making complicated University procedures appear easy, and her concern for my progress and welfare.

The support of this research by the Department of Computer Science at the University of Illinois, by the National Science Foundation under grant NSF MSC-82-05166, and by the Office of Naval Research under grant NOOO14-82-K-0186 is gratefully acknowledged.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Learning is a natural cognitive process. People and other living creatures learn almost constantly and yet very little is understood about what really goes on. Part of the interest in learning is a fascination with the diversity of learning situations. There is learning both with and without a teacher (e.g., being taught to identify poisonous vs. edible mushrooms and learning on one's own the characteristics of steak cooked different ways), and there is learning to perform a particular task (e.g., learning to tie one's shoe) and learning that reveals interesting relationships (e.g., learning types of automobiles, such as "foreign" and "domestic"). The study of machine algorithms that have an ability to learn is important for at least three fundamental reasons. First, machines that learn are needed to help solve difficult social, technical, and scientific problems of man in his universe. Second, by comparing the capabilities of artificial learning machines with human learning, cognitive scientists can investigate human intelligence as modeled by the machine learning algorithms. Third, the study of computer algorithms that learn provides insight into a genre of software that is different in many respects from other computer applications.

This thesis describes algorithms (along with their common underlying methodology) that have an ability to learn a taxonomy of descriptive concepts from examples without the assistance of a teacher. Each of the above reasons for studying machine learning algorithms is addressed. From the point of view of helping solve real problems, learning to describe collections of examples (so called *learning from observation*) is a powerful data reduction tool. We ourselves do this kind of reduction on data we receive, without even noticing it: rather than remembering all steaks we have ever eaten, we remember them as {rare, medium, well done} or as {juicy,

tough}, etc. The sample problem presented in Chapter 5 shows how the method was able to rediscover meaningful categories of soybean diseases. From the point of view of cognitive science research, building classifications of examples by machine and then comparing them to human classifications can reveal hidden algorithms involved in human intelligence. The sample problem in Chapter 7 has been used to study both human and machine learning, and as a basis for comparing inductive inference processes of the presented methodology with those processes involved in human learning. From the point of view of the study of computer algorithms, the path-rank-ordered (PRO) search described in Chapter 5 can benefit other computational tasks needing a specially adapted heuristic search through an immense space. These are the broad topics that will be explored in the thesis. Specific attention will be given to the algorithms for building a concept hierarchy to conceptually describe collections of examples. This is a process that has been named *conceptual clustering* by Michalski [Michalski, 1980b].

For the most part, the thesis is concerned with a single variety of conceptual clustering called *conjunctive* conceptual clustering, in which examples and conceptual clusters (classes) are described by conjunctions of predicates and relations over function/value pairs. The focus is placed on conjunctive forms of representation because such forms are commonly used in descriptive statements people generate to describe objects and concept classes. The conjunctive concept is an elegant form having both simplicity and expressive power. This provides a representation that is domain-independent and applicable to a very broad range of problems while at the same time being easy to manipulate by the application of a wide variety of generalizing and specializing transformations.

1.1. Four learning paradigms

The paradigm of "learning from observation" is but one of four widely recognized forms of learning (for both man and intelligent computer programs). The four forms are learning from instruction, learning by examples, learning by analogy, and learning from observation. In all

forms of learning, some external agent such as a teacher and/or the learning environment help direct the learning to varying extents. The minimum involvement of the external agent occurs in *unsupervised* learning where the teacher provides examples to consider and *biases*[1] the learner towards certain interpretations and away from others. In *supervised* learning, the agent is a teacher that determines how each example is to be used. The teacher differentiates between *positive* and *negative* events (or between events belonging to different classes) either by assigning a class to each example *a priori* or by serving as a consultant who states the class of any example when queried by the learner. The brief outlines that follow describe the unique characteristics of each learning paradigm.

1.1.1. Learning by instruction

In this paradigm of learning, the teacher provides complete *predigested* knowledge which the learner need only retain and apply. Memorizing rules for diagnosing and treating measles is an example of this kind of learning.

1.1.2. Learning from examples

In this paradigm of supervised learning, the teacher provides classified examples and counter-examples of the phenomenon. In some situations, the training examples are carefully selected by the teacher so as to be close to the conceptual border of the classes. Such examples are called *outstanding representatives* of the class. Near miss events just outside the class border are profitably used as negative events. In this form of learning, the learner must synthesize a mechanism for recognizing both examples and counter-examples, and to know which is which. Learning to diagnose a particular disease from a set of examples of the disease and a set of examples of other diseases and/or examples of a disease-free condition is an example of this type of learning.

---

1 Those preferences that are built into the method are called the *bias* of the algorithm [Utgoff, 1984]

Past work on this form of learning includes building descriptions for classes of blocks-world structures [Winston, 1975]; a family of inductive learning programs based on the A^q algorithm [Michalski, 1972, 1983]; the SPROUTER program which forms maximally specific conjunctive representations [Hayes-Roth, 1976; Hayes-Roth and McDermott, 1977]; the program THOTH which produces maximal unifying generalizations [Vere, 1975]; the program META-DENDRAL for discovering cleavage rules to explain mass spectrograms and give ideas about molecular structure [Buchanan et al., 1976]; and the program INDUCE for determining disjunctive structural descriptions of structured objects [Larson and Michalski, 1977; Larson, 1977]. A comparative summary of such methods is in [Dietterich and Michalski, 1981].

1.1.3. Learning by analogy

This type of supervised learning also involves examples provided by a teacher but the goal is to discover a transformation from one situation to another and to apply the transformation to new situations. Learning the properties of electric current flow from properties of fluid flow is an example of this type of learning. The paradigm of learning by analogy is not discussed in this thesis. Readers are referred to [Carbonell, 1983] and [Winston, 1979] for discussions of recent research in this area.

1.1.4. Learning from observation This type of unsupervised learning involves examples and possibly counter-examples, but unlike the paradigm of learning from examples, the goal is to synthesize a classification of the collection of positive examples by finding patterns which reveal an underlying conceptual structure in the collection. Conceptual clustering is one method for building classifications.[2]

Unlike the other three learning paradigms, the examples are not assigned initially to different classes. If counter-examples are provided, they are used in a way that is different from

2. Throughout the thesis, the term *classification* will refer to a classification resulting from the application or recursive application of conceptual clustering.

other learning methods. Any given counter-examples serve to bound the description space of possible classifications of positive examples (i.e., they keep the generated class descriptions focused on the kinds of entities of interest) but they are otherwise incidental to the classification building process. The learner alone must extract relevant information from descriptions of examples in a collection. The examples are related to each other in many ways but the nature of these relations is unknown to the learner.

An illustration of this kind of learning would be to build a classification of cars in a parking lot. An interesting outcome might be that a classification of cars by size might also classify them by model and country of manufacture. Since there is no "right" or "wrong" classification, the selection of a preferred classification from among many alternatives is based on an evaluation criterion that reflects the goal of the classification and on the bias of the classification building algorithm.

1.2. Thesis overview

The research described in this thesis deals with problems of learning from observation. In the chapters that follow, this type of learning will be further defined and contrasted with traditional approaches to show the fundamental differences between building traditional numerically-based taxonomies and concept-based classifications. Specific algorithms will be detailed for building classifications of examples (objects or situations) that are described by vectors of attribute values and those that are described by a more general predicate calculus type of representation.

Chapter 2 presents background information relating the broad area of *learning from observation* to other unsupervised learning approaches such as numerical taxonomy and other inference-based discovery systems. Chapter 3 describes the inference process called *conceptual clustering* and how it is related to the earlier general-purpose clustering technique called *dynamic clustering*. In Chapter 4, classification building via conceptual clustering is presented

as a process of inductive inference.

Several conceptual clustering approaches have been studied. Chapter 5 describes a method for building classifications for unstructured examples that are described by vectors of attribute/value pairs. Chapter 5 also presents sample classifications that resulted from the application of the method to a problem of rediscovering classes of soybean diseases. Chapter 6 presents a related but implementationally different method of conceptual clustering that handles examples described by an extended predicate calculus. The method is illustrated by an example problem of generating classifications of blocks-world objects. Chapter 7 describes a two-tiered approach that uses structural patterns found in a collection of examples to build a structural template with which a derived attribute-based representation is constructed. The derived problem description is processed using the attribute-based conceptual clustering algorithm of Chapter 5. This approach is illustrated with an example problem of generating a classification of some artistically conceived trains. Some results of studies by Medin on human classification building for the toy trains problem [Medin, Wattenmaker, Michalski, 1984] are summarized to relate the kinds of the machine-generated classifications with those produced by humans.

CHAPTER 2

BACKGROUND

2.1. Learning from observation

Problems of learning from observation involve unsupervised learning situations in which there is no teacher. The learning algorithm is presented with a collection of entities and it may do with them as it wishes. In living organisms, this kind of learning is motivated by the burden of learning massive amounts of data by rote. If strong patterns in the data can be found, then much of the information content can be retained more easily by remembering the strong patterns and their taxonomy rather than the individual data values. For example, having learned about automobiles, we remember the important characteristics and differences between classes of automobiles (by building taxonomies of automobiles) without specifically remembering each and every car we have seen. Learning from observation is a powerful and indispensable data reduction and knowledge acquisition paradigm.

2.2. Discovery systems

Learning systems which engage in unsupervised learning are called *discovery systems.* Classification-building (this work), theory-formation, and development-of-heuristics are examples of discovery tasks.

2.2.1. Previous conceptual clustering systems

There is little published work on artificial intelligence algorithms that perform conceptual clustering other than those of Michalski and Stepp [Michalski, 1980b; Michalski, Stepp and Diday, 1981; Stepp, 1984]. The remainder of the thesis is devoted to describing research based

on this approach. A related problem of building conceptual characterizations was studied by Stepp [Stepp, 1979]. In that work, inductive inference techniques were used to generate generalized characteristic descriptions that are maximally general while not surpassing a given *degree of generality*[1] threshold.

2.2.2. Clustering within a problem solving system

Problem solving often requires the efficient exploration of a vast state-space. Recently, clustering has come to play a role in reducing the vastness of the space to be explored. Rendell [1983] describes a *probabilistic learning system* (PLS) that performs heuristic searches through the space of possible solutions directed by the utility measure called *penetrance*[2]. The system contains a problem solving module guided by a control structure using a penetrance evaluation function. Different hypothetical solutions are clustered into regions with similar penetrance values. The clusters are hypercubes in the feature space and are generated using a divisive numerical clustering method in which the similarity measure is the penetrance score.

Clustering can be used in a variety of ways within a problem solving system. The PLS system demonstrates one important use of clustering in unsupervised problem solving by helping form solution regions composed of hypothetical solutions with similar utility scores. Another use of clustering in problem solving would be to reformulate the problem by subdividing it into sets of conceptually similar subproblems by conceptually clustering the problem goals or subgoals. This kind of clustering into conceptually similar entities (rather than numerically similar entities) is not performed by PLS.

---

1. A generalized description describes a set of points in the object representation subspace covered by the description. The subspace may contain points representing both some actual objects given with the problem, and a larger set of distinct hypothetical objects that are possible but not observed. The degree of generality of a description is measured as the ratio of the number of distinct hypothetical objects in the subspace to the number of actual objects in the same subspace.

2. The *penetrance* of a search tree is the ratio of the solution path length to the total number of states developed [Doran and Michie, 1966]

2.2.3. Theory formation systems

Current work in automated theory formation is represented by the BACON system [Langley, Bradshaw and Simon, 1983]. The BACON system (a family of programs) is a production system that discovers empirical laws. Equipped with a small set of heuristics, it detects regularities in numeric and nominal[3] data to formulate hypotheses about functional dependencies between variables and associations of values of nominal variables with intrinsic properties. The system processes empirical data and formulates summary descriptions at several levels of generality. For example, when given experimental data consisting of pressure, volume, temperature, and name of gas for different gasses over a range of volumes and temperatures, BACON discovers that pV/T is constant for each gas and that there is an intrinsic property "n" associated with the name of each gas which varies with the values of p, V, and T according to the equation $pV/nT = 8.31$ joule/degree.[4]

2.2.4. Systems for developing heuristics

Work in generating heuristics has led to the creation of the two systems AM and EURISKO [Lenat, 1983]. AM and EURISKO incorporate theory-driven discovery techniques and are specially designed to solve a unique class of problems. Their specialization immediately separates them from the general-purpose classification-building approach presented here. AM is a discovery program that can learn number theory concepts. Given a large number of mathematical concepts, represented in a frame-like fashion, and a still larger number of heuristic rules, AM sets out to look for regularities and interesting patterns in the concepts and in specific examples of the concepts. Local heuristic rules are used to update an agenda composed of a list of specific tasks to be performed. Some tasks explore certain properties of concepts; other tasks define new concepts.

---

3 Numeric data are measured on an ordered scale. Nominal data are measured on a nominal scale

4 From physics we know integer n is the ratio m/M where m is the mass of the gas and M is its molecular weight The actual intrinsic property is the molecular weight M When BACON is given only one sample of each gas it cannot detect the dependency on the mass m. When n is interpreted as the mass of the gas in moles, the ratio pV/nT is the universal gas constant 8 31 joule/mole-degree

The system operates in a select/execute loop in which the sole goal is to maximize the quality of the activities performed, i.e., to execute the tasks on the agenda with the highest *worth*. This heuristic-based approach was given a set of *prenumerical* concepts and proceeded to discover a variety of additional concepts (set membership laws, natural numbers, arithmetic operators, square-root, prime, etc.). The EURISKO program has been used to discover new heuristics and new kinds of representations, e.g., new slots for the frame-like concept representation it uses.

2.3. Feature extraction and noise reduction

Learning from observation can also be viewed as *feature extraction* or *attribute filtering*. With a (less than optimal) representation of any phenomenon there may be some attributes that are irrelevant to the situation at hand or whose values are noisy and inexact. The elimination of irrelevant attributes and the suppression of noise in the data can be considered as a transformation of the space of representations of the phenomenon. This space, called the *event space*,[5] is an n-dimensional space in which each point represents a possible combination of attribute values. A collection of examples can be mapped into this space giving rise to some distribution of marked points, one for each example (some events may coincide). By reducing the dimensionality of the space and by reducing the number of distinct values along single dimensions, the size of the space is made smaller. While there is no virtue in small event spaces *per se*, the reduction in size of the space leads to simpler descriptions of both examples and groups of examples located in the space. Some transformations cause more than one example to be translated into the *same* point in the new space. Such a point in the new space represents a *class* (or cluster) of examples rather than just a single object. Consider the following situations:

---

5 The terms *example* and *event* are used synonymously. Thus, the event space is the space in which each example can be represented distinctly.

2.3.1. Feature extraction

Suppose the examples are descriptions of certain animals (birds and mammals only, but this fact is not presented to the learner). Each individual animal is described by 30 attributes such as size, coloring, type of teeth, where it lives, what it eats, etc. The event space for such a problem is immense. By noting a pattern in the data that differentiates birds from mammals, one might propose a classification involving just those attributes that distinguish birds from mammals. Similarly, one might also notice a pattern that distinguishes those animals that live in or on the water from those that live on land. A projection of the event space onto just those relevant attributes would condense the space into four categories: birds that live on land, water birds, land-living mammals, and water-living mammals. The resulting event space would consist of just four points corresponding to four specific subspaces of the original event space. Only a fraction of the original attributes (or a new set of *derived* attributes) would be dimensions in the reduced event space. This process is a *feature extraction* performed over the original attributes. Traditional approaches to feature extraction are based on co-variances between attribute values. Such methods build a reduced set of attributes from linear combinations of original attributes such that the new attributes are mutually orthogonal.

2.3.2. Noise reduction

The following example will illustrate the ability to reduce noise through learning. Suppose the process being analyzed involves arrows shot from a bow hitting a target, with the attention placed on where the arrows hit rather than how they are shot. The complete event space would consist of positional information describing where the arrow hit the target. This two-dimensional space can be visualized much like the target itself, with the holes left by the arrows becoming the points in the event space representing observed examples of the phenomenon. Suppose that the archer is very skilled or replaced by a flawless machine. The arrows are launched identically for each example. Even so, ballistic scattering prevents them from hitting

the target in *exactly* the same place even though each arrow is an example of exactly the same phenomenon. Transforming the event space by reducing the number of possible position values of arrow impacts (e.g., by drawing concentric circles around the target center and measuring impact position only in terms of which region was entered) reduces the ballistic noise in the data. These kinds of data reduction and data analysis techniques have been practiced within the traditional methods of multi-dimensional scaling, feature extraction, and numerical taxonomy. A survey of traditional clustering techniques may be found in [Anderberg, 1973] and [Duda and Hart, 1973]. Complete coverage of numerical taxonomy is contained in [Sokal and Sneath, 1963]. The traditional approaches to data reduction problems have been based on numerical or statistical methods. Such approaches are appropriate for some problems, but not when a conceptual understanding of the generated classes is sought. In such cases the results are often difficult to interpret or apply.

In the following section the traditional method of numerical taxonomy will be briefly summarized to emphasize the differences between the traditional (non-conceptual) approaches and the inductive-inference-based (conceptual) approach that is the focus of subsequent chapters.

2.4. Numerical taxonomy

Numerical taxonomy is a technique that assembles examples into a tree-structured dendrogram such as the one shown in Fig. 2.1. A given distance metric (e.g., Euclidean distance) is applied between every pair of examples. The two examples which yield the lowest distance value are merged into one composite unit or group. The distance function is reevaluated over all remaining entities, with the newly formed group behaving as one entity. The distance function must thus be capable of measuring not only example-to-example distance, but also example-to-group and group-to-group distances. There are many ways to facilitate measuring distance relative to a group, given only an example-to-example distance function. The so called

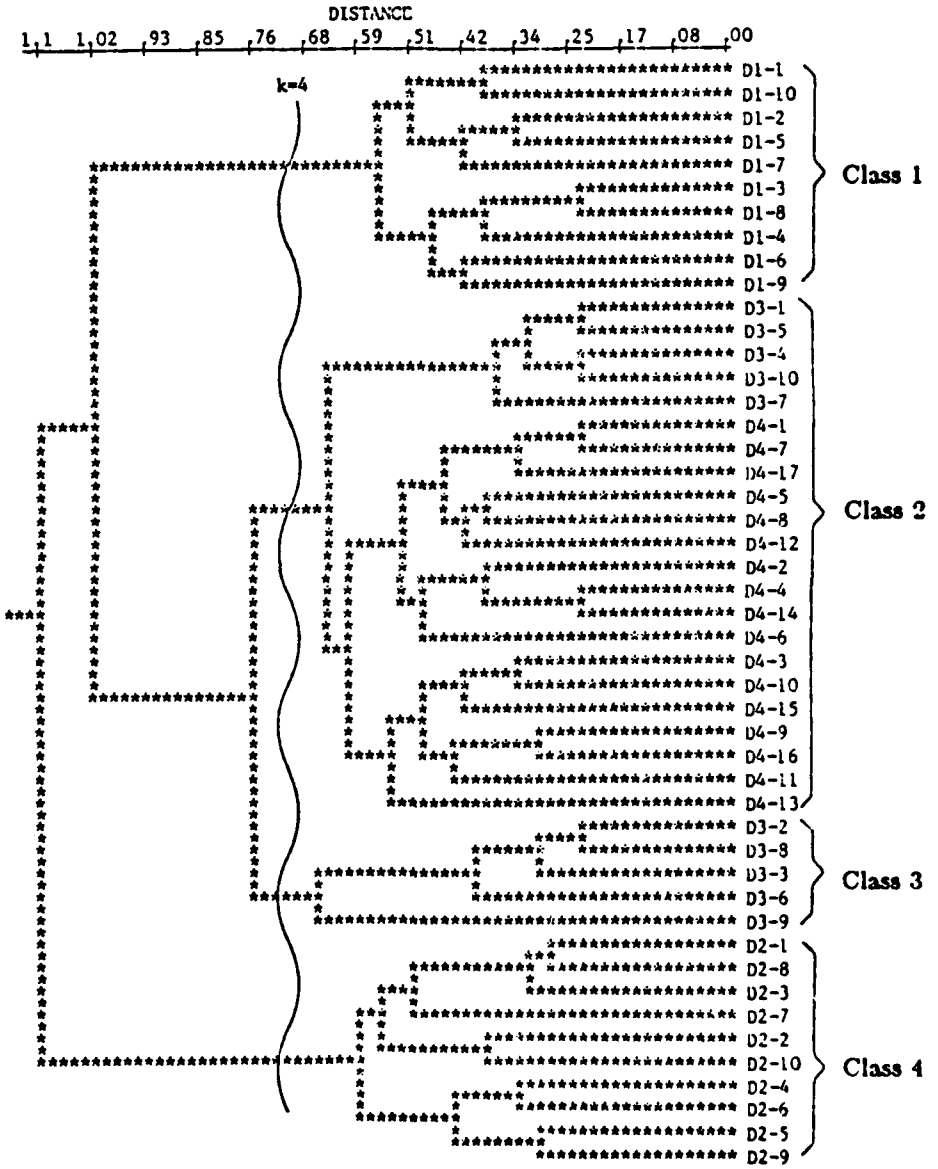
complete linkage methods measure group distance values by considering all distances from examples in one group to those in the other group and then taking the maximum distance. The *single linkage* methods do the same thing, but use the minimum distance. The so called *average linkage* methods take the arithmetic average of the distances. There is no universal agreement on which of the three group distance measurement techniques is best [Sokal and Sneath, 1963].

On each iteration of the merging operation, a group or example is merged with another group or example if the distance between them is the smallest. This continues until only one group remains. The history of the merging operations is represented as a tree (usually rooted at the left) in which each intermediate node is positioned horizontally according to an axis denoting the distances between the merged entities. A sample dendrogram is shown in Fig. 2.1. A dendrogram may be cut apart at some distance value to form a number of separate clusters of numerically similar entities. In Fig. 2.1 there is a wavy line showning where the dendrogram can be cut to form 4 clusters.

The main limitation of numerical taxonomy is that it is numerical rather than conceptual. The numerical techniques introduce many arbitrary biases and give rise to two fundamental problems:

1. The distance metric typically requires all attributes (dimensions) to be of ordered scale. Distance metrics for nominal scale attributes (for which the magnitudes of the differences in values are meaningless) are often based on the so called *simple matching score*[6] similarity measure. When a mixture of ordinal and nominal attribute types is present, there is a severe problem of how to combine the two different measures. This problem is usually handled by ignoring it, i.e., by mapping the nominal values into the integers starting with zero, and then pretending the scale is ordered rather than nominal. This gives nominal values unequal and arbitrary weights. All kinds of clever distance metrics

6 The simple matching score is measured by counting the number of corresponding attributes of two entities that have the same value



The wavy line shows where to cut to create four classes.

Figure 2.1. A dendrogram of diseased soybeans

have been devised (a brief summary can be found in [Michalski, Stepp and Diday, 1981]) but the problem of handling both nominal and ordered attributes in one distance metric has not received a satisfactory solution.

2. The attributes used to describe entities may have substantially different ranges of values, even if they are all of the same scale. If raw numerical values are used, then there is an implied arbitrary weighting that corresponds to irrelevant factors such as unit size of the measurements (e.g., one attribute is measured in feet and one is measured in inches; or worse yet one is measured in feet and another is measured in pounds). Statistical techniques such as normalization or standardization may be used though such techniques merely substitute one arbitrary weighting for another one. How to choose weights is an unanswered question in numerical taxonomy. In many instances, the weights are assumed to be 1.0 for lack of any better wisdom.

Numerical taxonomy has two redeeming characteristics: it is fast, and it is useful for those classification-building problems in which distance is a relevant parameter. Even when the technique is appropriate, the output such as the dendrogram shown in Fig. 2.1 is devoid of interpretation. When a data analyst partitions examples by way of a dendrogram, the significance and description of each class must be determined by external means, generally through the perceptual abilities of the analyst himself. The mathematical nature of the numerical taxonomy process cannot produce patterns of objects that (except by coincidence) match concepts people might use to describe them.

CHAPTER 3

THE CONCEPTUAL CLUSTERING PARADIGM

This chapter discusses the fundamental properties of conceptual clustering by focusing primarily on those facets which differentiate conceptual clustering from other clustering techniques. Since the thesis is concerned with *conceptual* clustering, the chapter begins with some notions about concepts and how they are different from names for things. The next section discusses the general process of building a concept-based classification of examples using conceptual clustering. The clusters (classes) formed are described by concepts and are composed of examples that are similar, not by a conventional mathematical measure of similarity, but by a *conceptual* similarity measure introduced by Michalski [1980b] called *conceptual cohesiveness*.

Following the introduction of conceptual cohesiveness, two sections present the three basic approaches to non-conceptual clustering and then briefly describe the so-called *dynamic clustering* approach that is part of the framework on which the method of conceptual clustering is built. Another fundamental part of the underlying framework is a reliance on various kinds of background knowledge. The role of such knowledge in clustering is described in Sec. 3.6. Finally, the chapter ends with three variations of the clustering paradigm that handle one dimensional clustering, clustering of weighted examples, and clustering in the presence of negative examples. These topics in Chapter 3 are preparatory to the introduction of the underlying inductive inference algorithms which starts in Chapter 4.

3.1. Concepts

There is still some debate in philosophical discussions about just what is meant by the term *concept*. A clear distinction exists between "Joe's telephone number" as a particular 7-digit

number and "Joe's telephone number" as a description of some object class. Although one could argue that even "555-1234" (a particular 7-digit number) is an "elementary concept" because it describes the object class that contains itself as the only object, such usage is extreme and limiting. Without loss of generality, the term concept will be taken to mean a description of a class of potentially many entities.

The underlying distinction between names of things and concepts is that concepts come with certain properties of entities belonging to the concept class. Rather than "Joe's telephone number" being just a name for a certain number, the concept "Joe's telephone number" has properties that do not depend on what number "Joe's telephone number" happens to be. Some of the properties are:

1. The number will be accepted by the phone system.
2. When called, Joe is likely to answer.
3. When called, a message may be given to Joe or left for him to receive later.

A concept exists whether there is a phrase in the English language for it or not. If denoted by a single phrase, it is possible to construct an equivalence between the assertion that an object is such a concept and a statement (usually conjunctive) of the properties exhibited by such an entity. For example

$$\forall\ \psi,\ \text{is\_Joe's\_telephone\_number}(\psi) \Longleftrightarrow$$

$$\text{accepted\_by\_system}(\psi) \wedge \text{likely\_to\_answer}(\text{Joe},\psi) \wedge \text{send\_message\_to}(\psi,\text{Joe})$$

represents, in predicate calculus notation, the equivalence between the named concept "Joe's telephone number" and its property set. Concepts not previously defined in the English language may be represented by stating just the property set, i.e., by giving a logical statement such as the right hand side of the above equivalence.

A powerful conceptual form which is frequently used by people when describing concepts is the conjunctive statement. In the research described in Chapters 4 to 7, all concepts are

conjunctive statements. Concepts will be represented in either extended propositional logic or extended predicate calculus using subsets of *Annotated Predicate Calculus (APC)* [Michalski, 1983]. Thus, in this thesis, a concept is a conjunctive statement in APC. The described *conjunctive* conceptual clustering is a process that generates conjunctive-form concepts.

3.2. Classification building

A *class* is composed of entities that are instances of some concept. A *classifier* is a process that maps every possible entity into one or more classes, according to a set of *class recognition rules* or *class descriptions*. Both class recognition rules and class descriptions can be written in the form

$$\text{CONDITION} \Rightarrow \text{CLASS-NAME}$$

where CONDITION is a statement that is satisfied by every example of the class whose name is CLASS-NAME. The condition part of a class recognition rule is usually maximally general under the constraint that it remains unsatisfied for any example of another class. The condition part of a class description is maximally specific under the constraint that it is in acceptable form, e.g., the form may be limited to that of conjunctive concepts—this would necessitate a certain amount of generalization to avoid disjunction. A class recognition rule contains *discriminant* features that are predicates and/or functions which take different non-intersecting value sets in different classes. Class descriptions contain *characteristic* features that have value sets representing the observed values for examples of the class. Value sets of characteristic features of different classes may intersect.

For traditional classifications, the class-recognition rule may be a discriminant function which defines a boundary separating one class from another (e.g., a hyperplane). For conceptual classifications created by conceptual clustering, we are interested in class descriptions that are concepts. Conceptual class recognition rules may be obtained by maximally generalizing class descriptions, i.e., by removing the characteristic features and keeping only the discriminant ones.

A *classification* of a given set of objects is a partitioning of the set into k subsets according to a certain classifier. In the thesis no attention is given to classifications based on arbitrary partitionings of objects, but rather to cases where the partitioning is based on concept class membership. In such cases of *conceptual classification* building, both the partitioning of examples and the classifier are generated by the classification building process. A *hierarchical* classification is obtained by recursively building hierarchical or non-hierarchical classifications for each of the subsets of objects resulting from a top-level (or the previous level) classification.

The process of building a conceptual classification for a collection of objects involves synthesizing an optimized classifier using conceptual clustering and using it to arrange the objects by class. The classification is evaluated according to the evaluation criterion (*goal*) and if alternative classifications are possible, the one with the best evaluation score is selected (i.e., the one that most nearly attains the goal of the classification is selected).

When objects are arranged into groups by building conceptual classifications, the objects in one group are not merely *similar* to each other in a conventional way, but have high *conceptual cohesiveness* (defined below) because they are all instances of the same concept. The idea of conceptual cohesiveness [Michalski, 1980b] is fundamental to the theory underlying conceptual clustering.

3.3. Conceptual cohesiveness

In traditional clustering approaches, the similarity between objects A and B depends only on their attribute values. Measures such as reciprocal Euclidean distance and the so called *simple matching score* (the number of attributes having the same value for both objects) are functions that are based solely on the values of the attributes of the compared objects, i.e., the similarity is some function of the attributes of A and B:

$$sim = f(A,B)$$

To permit the similarity measure to take into account patterns which can only be detected when

considering more than two objects at a time, the *environment* of the compared objects (i.e., attribute values for surrounding objects) is also used. This *context sensitive* measure of similarity is a function of three parameters:

$$sim = f(A,B,E)$$

where E denotes the descriptions of all objects in the environment of objects A and B. An example of this kind of similarity measure is the reciprocal of "mutual distance" [Gowda and Krishna, 1978]. To calculate the mutual distance between objects A and B, all objects are ranked according to the Euclidean distance to A (the closest object is ranked 1) and then according to the distance to B. The mutual distance from A to B is the sum of the rank value of A with respect to B and the rank value of B with respect to A.

Taking neighboring objects into consideration solves some clustering problems, but in general is not sufficient. The difficulty lies in the fact that the above types of similarity measures are still *concept*-free, i.e., they depend only on the properties of individual objects and not on any concepts that might be useful to characterize object configurations. Consequently, methods that use concept-free measures are fundamentally unable to capture the gestalt properties of object clusters, i.e., properties that conceptually characterize a cluster as a whole and are not derivable from properties of individual entities. In order to detect such properties, the system must be equipped with the ability to recognize configurations of objects that correspond to certain concepts.

This idea is the basis of conceptual clustering. The *conceptual similarity* between two objects A and B, which is called the *conceptual cohesiveness* of A and B, depends not only on those objects and surrounding objects in environment E, but also on a set of concepts C that are available for describing A and B together with other objects [Michalski, 1980b]. Thus, the conceptual cohesiveness between two objects A and B is a four-argument function

$$sim = f(A,B,E,C).$$

3.4. A taxonomy of clustering techniques

Clustering techniques come in three basic forms: *agglomerative* techniques such as numerical taxonomy that repeatedly join separate entities and small groups together, forming larger and larger groupings until only one group exists; *divisive* techniques that repeatedly break an initial single starting group into smaller and smaller chunks until only single entities exist in each chunk; and *direct*[1] techniques that are given the number k and proceed to construct k clusters by arranging (and iteratively rearranging) the entities into k groups.

The optimal number of clusters is determined differently in each approach. For divisive cluster formation, a stopping criterion determines when to cease further cluster division at some optimum stage prior to obtaining clusters composed of single entities. For agglomerative clustering, the entire agglomeration is completed while producing a dendrogram such as the one shown in Fig. 2.1. The dendrogram is then cut into clusters according to a between-group similarity threshold. For direct clustering, the number of clusters is given. An optimal number of clusters to form can be found by performing direct clustering for a range of values for k, and then selecting the clustering that is best according to an overall criterion that is valid over clusterings with varying numbers of clusters.

Agglomerative techniques merge together individual entities and small groups according to locally applied similarity scores. This makes such techniques inappropriate for finding global patterns in the data which may only be revealed by considering larger groups of objects together. Divisive techniques split a large group into two smaller groups. This leads to a hierarchy of object groups. One problem with divisive clustering is that the criterion for how to split a group may be sensitive to global patterns in the collection of objects. A poor split early in the algorithm cannot be overcome at later stages. Direct techniques are best suited for optimizing a global criterion.

---

1 Some writers use the term *non-hierarchical* to describe clustering techniques that are given the number of clusters to form *a priori*. That term is misleading in the context of the methods presented here which build hierarchies of classes by recursively applying a direct clustering technique.

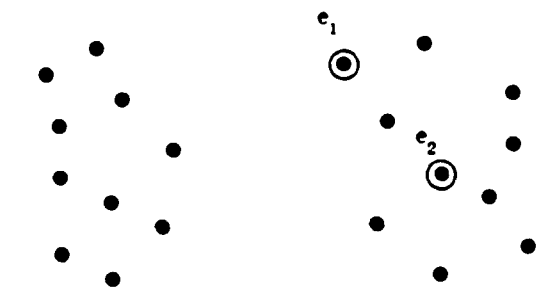
The approach taken in the conceptual clustering method described in this thesis is to perform direct clustering to partition a group of entities into classes according to a set of conceptual class descriptions. The classes may be further subdivided by recursively reapplying the conceptual clustering algorithm to each class. The result is a conceptual hierarchy. The direct clustering scheme used is an adaptation of an approach known as *dynamic clustering*.

3.5. Dynamic clustering

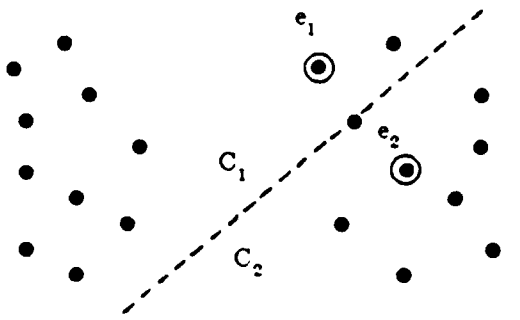
The dynamic clustering method [Diday and Simon, 1976; Diday, 1978] is a direct clustering technique that finds clusters iteratively by alternately applying a *representation function* and an *allocation function* until a stopping criterion is reached. The technique is related to earlier techniques such as *K-means* [MacQueen, 1967] and *Center adjustment* [Meisel, 1972]. These two techniques use reciprocal Euclidean distance as the similarity metric and represent clusters by their centers of mass. They do not perform conceptual clustering.

In dynamic clustering, the representation function forms a *representative set* for each of the k clusters. A typical representative set for a cluster would be one or more objects from the cluster that best illustrate the characteristics of all objects in the cluster, e.g., an object that is conceptually *central* among the objects in the cluster. One way to measure the conceptual "centralness" of objects is by summing an attribute-value difference measure applied between all object pairs in the cluster. (Such a measure, called the syntactic distance between two concepts, is defined in Chapter 4.)

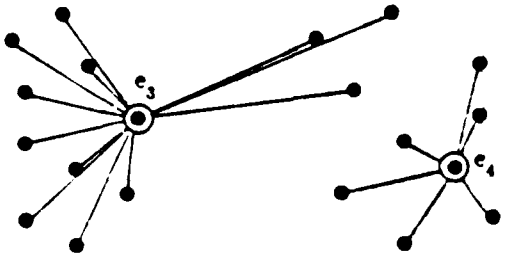
The allocation function partitions the set of objects to be clustered according to object similarity to representative objects. These processes are illustrated in simplified form by the diagrams shown in Fig. 3.1. In panel (a) one representative object (the seed event) is selected for each class. One feature of dynamic clustering is that the initial choice of seed events is not very important. As shown in panel (a), the initial seed choice with both examples fairly close together in the event space is quite poor. Even so, a good clustering is established in panel (d) after



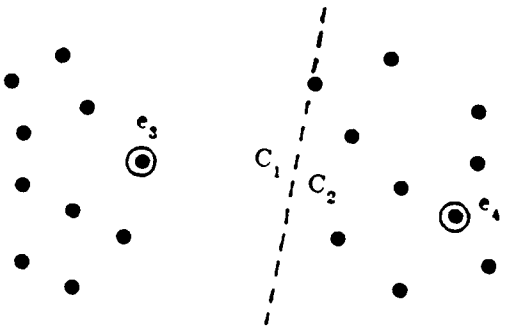
(a) Obtain initial representation. Two seed events e_1 and e_2 are selected at random. (Seed events are circled.) The similarity function is reciprocal Euclidean distance.



(b) Allocate events to classes. Points most similar to e_1 are placed into class C_1. Points most similar to e_2 are placed into class c_2. The dashed line shows the boundary beween class spaces.



(c) Form new representation. A new representative is selected for each class. The most central event having the minimum sum of distances to other events in the same class is selected. The thin lines mark the distances summed for the selected events.



(d) Allocate events to classes. Allocation is performed as in snapshot (b) using the new seed events. Note that after just two full iterations the method has discovered the strong pattern of two groups of points. Subsequent iterations of the algorithm will indicate the same two class solution.

Figure 3.1. An illustration of dynamic clustering with two classes.

completing only two iterations. In panel (b) the examples are allocated to classes on the basis of similarity to the seed of the class. In the (non-conceptual) illustration, the classes are delineated by the hyperplane border shown as the dashed line. Panel (c) illustrates the process of selecting new seed events. Sums of distances between all events that were in class C1 in panel (b) are calculated. The event shown (labeled e_3) is the one whose sum of distances was lowest. It is one of the two seeds for the next iteration. From the events which were in class C2 the other new seed event e_4 is selected in a similar fashion. In panel (d) the two new seed events are used to again allocate all events to classes, performing the same process used in panel (b). The class regions C1 and C2 correspond to the spatially apparent clusters of points. Further iterations will maintain this same partitioning, although the hyperplane will move closer to the center of the space.

3.6. The role of background knowledge

It can be observed that when people create a classification of observations or partition a system into subsystems, they employ knowledge about various concepts and relationships on attribute value sets that are relevant for describing the observations or the system. This knowledge which is independent of the descriptions of the examples will be called *background knowledge*. It can provide specialized generalization transformations over attribute values and an ability to construct new descriptors (attributes, functions, or predicates) from those that are initially given. Both of these applications of background knowledge to the construction of classifications add a powerful capability that humans innately possess but which few inference programs do.

Background knowledge also solves a nagging classification building problem: that of deciding which of the multitude of possible classifications for the same example set are best. Since there are no *right* or *wrong* classifications, the term "best" is hard to define. Part of the role of background knowledge is to specify the *bias* of the classification building process, i.e., to

provide an appropriate mechanism to evaluate classifications and select one which best meets the goal criterion. The term *bias* denotes the cumulative influences of an inference process on the generation and selection of alternative hypotheses [Utgoff, 1984].

During classification generation, the incomplete hypotheses are evaluated. The best ones survive to become final results while computational resources are deprived from those classifications which fall short. In the work described in this thesis, tactical knowledge of how to evaluate classifications is incorporated in the algorithms through the Lexicographical Evaluation Functional with tolerances *(LEF)* [Michalski, 1980b], explained in chapters 5 and 6.

Another use of background knowledge is to help determine which features are relevant to the classification problem. Event descriptors can be divided into *initial descriptors* and *derived descriptors*. The initial descriptors can be further divided into those that are relevant and those that are irrelevant to the problem. In some problems, the complete set of relevant descriptors is unknown and not necessarily provided as initial descriptors. A solution can still be obtained in such cases if background knowledge can be used to derive relevant descriptors from those that are initially given. When considering both initial and derived descriptors, it is possible to distinguish three categories of descriptors, as given below.

Initial descriptors

Initial descriptors are those predicates and attributes of objects and object subparts that are present in the given object descriptions. For physical objects the descriptors will be size, color, location, connectivity relationships for object subparts, etc.

Descriptors derived by logical inference

These descriptors are predicates and functions obtained by the application of general and problem-specific inference rules to the initial descriptions of the objects. In this work, inference rules consist of a condition part and a consequence part. Whenever an object description matches the condition portion of a rule, the consequence portion is applied to

the object description. The consequence may be composed of new predicates and functions to be asserted or arithmetic expressions that are evaluated. In both cases, the new descriptors (unless already present) are appended to the object description and become available as attributes that are potentially relevant for building classifications of the objects.

Descriptors derived by special computations, experiments or devices

These descriptors are obtained from the initial descriptors by the application of specialized descriptor generation procedures, by running experiments, or by activating some external device. Examples of such descriptors include "the number of object subparts," "the number of subparts with some specific property," "the number of different values observed for an attribute," and "properties common to all subparts."

Let us consider a simple example. Suppose that we are observing a typical restaurant table on which there are such objects as food on a plate, a salad, utensils, salt and pepper, napkins, a vase with flowers, a lamp, etc. Suppose a person is asked to build a meaningful classification of the objects on the table. One way to create a classification is to perform the following chains of inferences:

- salt and pepper are seasonings
  seasonings are added to food
  seasoned food is something to be eaten
  things which are to be eaten are edible
  salt and pepper are edible

- salad is a vegetable
  vegetables are food
  food is something to be eaten
  things which are to be eaten are edible
  salad is edible

A similar chain of inferences applied to "meat on a plate" or "cake on a dessert plate" will also lead to the concept "edible." On the other hand, a lamp is not food and is therefore not edible. A vase containing flowers is not food and is therefore not edible. Consequently, with the aid of

pertinent background knowledge inference rules, one meaningful classification of objects on the table is the division of objects according to the classifier "edible vs. inedible."

The problem this example poses is this: suppose we are given descriptions of objects on the table in terms of their physical attributes and we want to have the program create, on the basis of these descriptions, the above classification of edible and inedible objects. Such a classification based solely on original attributes is quite unlikely, since neither the "edible" predicate nor any equivalent attribute is an easily observed physical attribute of an object. It is necessary to derive descriptors such as "edible" using available background knowledge. A program that could classify objects on a table as edible or inedible would have to be equipped with (among other things) the above inference rules and with the ability to use them.

3.7. Special clustering situations

One-dimensional clustering

A special situation exists in the case of one-dimensional conceptual clustering. When clustering the values of a single attribute, the resulting concepts must be generalized values in the domain of the attribute (e.g., intervals in the case of a linearly ordered domain). For some attribute domains, background knowledge describing preferred generalization transformations can be brought to bear on the problem. For example, the attribute *shape* may have a domain containing the values *square*, *triangle*, *circle*, *ellipse*, and a generalization hierarchy that provides *polygon* as a generalization of *square* and *triangle*, and *curved* as a generalization of *circle* and *ellipse*. This type of generalization transformation is called *climbing the value hierarchy*. It will be discussed again in Chapter 4.

Clustering repeatedly occurring or weighted examples

In some situations, objects in a collection are not considered equally important. Such differences in objects are relevant when building a classification that tries to reveal strong

patterns in the data. Certain object descriptions, which occur repeatedly in the collection when others occur only once, should be treated differently. Furthermore, since repeating objects are conceptually identical, it is not necessary to consider them more than once. Thus, multiple occurrences of objects are merged into one object with appropriately increased importance. This notion of importance is handled by assigning to each object a unit *weight*[2] and by automatically editing each collection of objects to eliminate duplicates and arithmetically combine weights. The use of object weights in the evaluation of the degree of fit between a set of objects and an associated concept is discussed in Chapter 5.

Clustering with negative examples

Building a classification by conceptual clustering is normally an unsupervised process (without a teacher). A variation of the basic method can utilize a teacher that circumscribes the the collection of entities being classified with well-chosen negative examples. The goal of classification building in such a situation is to find a set of concepts which form a classification for the positive examples and which do not cover any negative examples. For example, if one were creating a classification of automobiles, a teacher could guide the process away from building classes that might also cover trucks by providing some examples of trucks as negative examples to the classification building process.

---

2 Provision can be made for arbitrary user-defined initial weights for each object rather than unit weights

CHAPTER 4

CLASSIFICATION BUILDING AS INDUCTIVE INFERENCE

This chapter presents conceptual classification building as a process of inductive inference. In Chapter 3 the general algorithm of dynamic clustering was presented. A different view of the clustering process is outlined in Fig. 4.1. This view presents conceptual clustering as a process of *repeated discrimination*, i.e., repeated application of inductive inference in the form of *learning from examples*. In order to apply inductive inference techniques, a representation scheme for both examples and classes of examples is needed which supports a wide variety of generalizing and specializing transformations. This chapter presents a framework for conceptual clustering

Given: k, the number of clusters desired
E, the collection of observed examples
B, the available background knowledge
H, the classification goal
S, the stopping criterion or resource limit

Apply background knowledge B to examples E.
Repeat until stopping criterion S is met.
Form a partial partition hypothesis by assigning one or more examples (seeds) to each of k classes.
Use inductive inference (learning by examples) to generate class descriptions which optimize the classification goal H.

Figure 4.1. The basic conceptual clustering algorithm of repeated discrimination.

based on inductive inference techniques. The method used has some similarity to dynamic clustering but is carried out by applying inductive learning algorithms and transformations to examples described using a representation scheme based on propositional logic and predicate calculus.

The three sections of the chapter describe the inference framework with respect to an associated representation language, some useful generalizing and specializing transformations, and a process for building classifications by repeated discrimination. This framework for inference-based conceptual clustering is used later to build three methods for generating classifications. The inductive inference techniques of this chapter used with an extended propositional logic representation (i.e., attribute/value-set pairs) form the basis for the method presented in Chapter 5 for building classifications of unstructured examples described by vectors of attribute values. The same inductive inference techniques combined with an extended predicate calculus representation form the basis for the method presented in Chapter 6 for building classifications of structured examples represented by statements in predicate calculus. A combination of the two methods is presented in Chapter 7. That method uses a structural template to transform a problem with structured examples into derived subproblems with unstructured examples.

4.1. Problems of partitioning

The problem of how to build a classification for a collection of objects is decomposed into two parts: "how many classes (clusters) are best?" and "what are the class descriptions?" Both questions are to be answered with respect to the classification goal. Since class descriptions cannot be obtained until the number of classes is known, and since choosing the optimal number of classes requires the ability to evaluate the classifications produced (as determined by class descriptions), there is no straightforward way to answer both questions optimally.

A tractable approach is to evaluate and compare the best classifications produced for a range of numbers of classes. Let the minimum and maximum numbers of classes be k_{min} and k_{max}, respectively. For each value of k between k_{min} and k_{max}, the best classification involving k classes is determined. The value of k associated with the best overall classification determines the optimal number of classes to form.

By constraining k to lie between k_{min} and k_{max} (with k_{min} and k_{max} fixed arbitrarily) it is no longer possible to generate the optimal classification if the best value for k happens to lie outside the interval considered. By constructing a hierarchical classification rather than a one-level classification, k_{max} can remain small and yet an unlimited number of classes may be defined as leaves in the hierarchy with only k_{max} branching at each level. The k_{max} branching constraint of the hierarchical classification may for some problems preclude the construction of the optimal classification.

Let us presume the above approach for finding the optimized value for k. The remaining task is to generate, for each trial value of k, the classification involving k classes that optimizes the classification goal. The procedure to do this involves iterative inductive inference of *repeated discrimination* (repeated learning from examples).

To see how this works philosophically, keep in mind that the result is a set of k generalized descriptions that are mutually disjoint, cover all positive examples in the collection, and are optimal (or near optimal) with respect to the classification goal. Because the descriptions are mutually disjoint, they can be used to partition the objects into k object classes, such that the i^{th} description covers only objects in the i^{th} class.

Now suppose the correct partition exists but that the class descriptions are unknown. This is the traditional *learning from examples* problem[1] in which there are k classes and it is desired to find k optimized discriminant descriptions for the classes. Procedures to produce such quasi-

1 Actually, a modified learning from examples paradigm is used that generates *class description* rules. The regular paradigm of learning from examples generates rules of the more general *class recognition* type.

optimal discriminant descriptions are well known [Michalski, 1980a, 1983]. Getting the partition of objects that optimizes the classification goal remains the crux of the matter.

Finding the correct partition is computationally difficult because there are so many possible partitionings of m objects into k classes. The generate-and-test approach cannot succeed in cases such as this unless only a tiny subset of possible partitionings are investigated. Unfortunately, heuristics to select partitionings likely to score well with respect to the classification goal are unknown.

One solution to the partitioning problem comes from inductive inference. Remember that a partition of the objects is being sought in order to use a *learning from examples* approach to generate class descriptions. Learning from examples is a robust procedure that can produce nearly correct class descriptions from incomplete sets of examples. This property can be used here to help partition the objects into classes.

If a partioning of the objects were available, the objects in one group would constitute a set of examples of a class and the union of examples in other groups would constitute negative examples of the class. An application of the paradigm of *learning from examples* would generate a description of the class covering all (positive) examples and no negative ones. The above procedure would be performed a total of k times to generate descriptions for each class, with a different group in the partition constituting the positive example set on each iteration.

Suppose instead, that only k objects (called *seeds*) are used, one selected somehow from each class. A single object in one class would be the only positive example while the k-1 seed objects in the other classes would form the set of negative examples. When *learning from examples* is applied in this situation, the resulting descriptions will likely be more general than before when all examples were used because there are fewer negative events. Strong patterns in the collection of objects will still emerge in the generated class descriptions because each class continues to be represented by one example.

The above seed-based approach is actually run in reverse. If k examples can be selected such that each example comes from a different one of the (unknown) classes that are optimal with respect to the classification goal, then the generated class descriptions should, in a generalized way, reveal the class structure. The algorithm is shown in Fig. 4.1. The hard part is the selection of seed objects.

The desired result of such an algorithm is a conceptual clustering (a partitioning of all objects with corresponding disjoint class descriptions) that optimizes the classification goal in the special situation that the seed examples are "correctly" selected, one from each class. Of course the "correct" choice of seed examples is unknown. Selection heuristics based on the previous hypothetical partitioning have been designed, based on the dynamic clustering method described in Chapter 3. Repeated application of the inner loop of the algorithm (Fig. 4.1.) using the selection heuristic tends to select improving choices of seed examples, and hence, better conceptual clusterings. With a stopping criterion to limit the expenditure of computational resources, near optimal class descriptions are generated. Further details of this method are given later in Sec. 4.4.

The heart of the above process is inductive inference over descriptions of examples. This implies having a suitable underlying representation language and associated generalization transformations.

4.2. Representation language

In order to build a classification of objects, it is necessary to have an adequate language for describing objects as well as classes of objects. This requirement suggests the use of a predicate calculus or some modification of it. Here, a language created by Michalski [1983] called *Annotated Predicate Calculus (APC)* is used. APC is an extension of predicate calculus that uses several novel forms and attaches an *annotation* to each predicate, variable, and function. The annotation is a store of information about the given predicate or atomic function such as

the definition of the function's value set. In addition to all forms found in predicate calculus, the APC language also employs a special kind of predicate called a *selector*. A simple selector is in the form:

[atomic-function REL value-of-atomic-function]

where REL (relation) stands for one of the symbols $= \neq < > \leq \geq$. An example of such a selector is

[weight(box) > 2kg]

which means "the weight of the box is greater than 2 kg." "Box" is the argument (often existentially quantified) of the single-argument function "weight." For multi-argument functions and predicates, the notation f(a,b,c, ...) is used to denote a function having positional significance of its arguments. When argument position is irrelevant, the notation f(a.b.c. ...) is used. With two-place predicates, the notation p(a.b) denotes an anti-symmetric predicate while p(a.b) denotes a symmetric predicate for which p(a.b) = p(b.a).

A more complex selector may involve *internal disjunction* or *internal conjunction*. These two operators apply to terms rather than to predicates and are illustrated by the two corresponding examples:

[color(box) = red $\vee$ purple] "the color of the box is either red *or* purple."

[color(box1 & box2) = red] "the color of box1 *and* box 2 is red."

The meaning of the internal disjunction operator is defined by

$$[f(x)=a \vee b] \iff [f(x)=a] \vee [f(x)=b]$$

and the meaning of the internal conjunction operator is defined by

$$[f(x \,\&\, y)=a] \iff [f(x)=a] \,\&\, [f(y)=a].$$

Selectors can be combined by standard logical operators to form more complex expressions.

Background knowledge can be expressed as a set of APC implicative rules:

CONDITION $\Rightarrow$ CONSEQUENCE

where CONDITION and CONSEQUENCE are conjunctions of selectors. If CONDITION is

satisfied, then CONSEQUENCE is asserted. To illustrate the implicative statement, consider the assertion "vegetables are food" from the example in Chapter 3. It can be expressed:

$$[\text{is-vegetable(object1)}] \Rightarrow [\text{is-food(object1)}]$$

An alternative way to express it is

$$[\text{type(object1)} = \text{vegetable}] \Rightarrow [\text{type(object1)} = \text{food}]$$

In this latter expression "vegetable" and "food" are treated as elements of the tree-structured domain of the attribute "type." This implication expresses a generalization inference rule called *climbing the generalization hierarchy* (the domain of the attribute "type" is assumed to have a tree structure). Further details on the APC language are given in [Michalski, 1983].

A conjunctive statement in APC can describe conceptually a single example, or a class of examples, depending on its generality. An equivalent view of such a conjunctive statement is as a cover for some part of the feature space in which all examples are described. (This space is called the *event space*.) A statement describing a single example covers (is satisfied by) only one point in the space. A more general statement would cover (be satisfied by) many points. Covered points that correspond to given examples are called *observed events*. Points which have no corresponding example in the given collection of examples are called *unobserved events*. For a given set of observed events, the best fitting statement that covers the events with minimal generality is the one that covers the fewest unobserved events.

A class of examples can thus be defined in two equivalent ways. One way is by giving a set-theoretic enumeration of all events (both observed and unobserved) that are in the class. The other way is by giving a conjunctive statement that is formulated such that it is satisfied only by examples in the class. This dual interpretation does not hold for all possible arrangements of examples since not all combinations of events can be described precisely by a single conjunctive statement.

Given any set of examples (observed events), covering theory shows that there exists a conjunctive cover of the entire set of observed events in all cases only if it is permissible to cover

some additional unobserved events [Michalski, 1974]. Such a cover is called a *complex*. It is a dual form entity that has both logical and set-theoretic properties. A complex is written as an APC conjunctive[2] statement. The logical interpretation of a complex is a conceptual description that is satisfied by all examples in its concept class. The set-theoretic interpretation of a complex is a subset of points in the event space. Any examples represented by a point in the covered subset belong in the described class.

A complex differs from an arbitrary class of examples (an arbitrary collection of points in the event space) in that a *complex* always supports dual interpretations, i.e., an equivalent conjunctive-form APC statement is always available. The notion of a complex will be useful when it is desirable to talk about only those subsets of points (examples) in the event space for which there exists an equivalent conjunctive-form APC statement. The term *ℓ-complex* explicitly denotes the logical interpretation of a complex while the term *s-complex* explicitly denotes the set-theoretic interpretation. Both interpretations are implied simultaneously by the term *complex* without a prefix.

4.3. Representation transformations

The process of conceptual clustering (as presented here) can be described as a transformation from descriptions of objects as points in a multi-dimensional event space to descriptions of classes of objects described in a one-dimensional *class space*. There will be k such classes that can be labeled C_1, C_2,...,C_k. It is assumed here that the classes are described by conjunctive statements α_1, α_2,...,α_k. A conjunctive statement α_i containing p_i conjuncts has the form

$$\alpha_i: \qquad S_{i1} \wedge S_{i2} \wedge \ldots \wedge S_{ip_i} \Longleftrightarrow C_i$$

Concept C_i is defined by the conjunction of properties S_{ij}, where the S units are *selectors* in APC.

---

2. A conjunctive-form statement may contain selectors that employ internal conjunction and internal disjunction. A conjunctive-form statement may not involve the (external) disjunction of selectors.

When all α_i are mutually disjoint, each α_i describes a transformation from p_i-space into *class space*. Normally, a problem is not originally stated in p_i-space and many other transformations are applied to go from the initial representation space to p_i-space. Some transformations may reduce the dimensionality of the description space; others may reduce the number of relevant values along some dimension; some transformations may construct one set of descriptors (dimensions) from another. Although there are set theoretic interpretations to all transformations used, it is the conceptual interpretation within the APC representation language that is most interesting. Many such transformations are described below. Most of these transformations (inference rules) were developed by Michalski [1983].

Dimension-reducing transformations (generalizing transformations)

- *Dropping condition rule:* a conjunctive description can be generalized by removing one or more conjuncts.

Relevant value set transformations (generalizing transformations)

- *Adding alternative value rule:* a selector conjunct [f(x)=a] can be generalized by adding one additional value b (or more) in order to form the selector [f(x)=a∨b]. The composite value "a∨b" replaces the values "a" and "b" and thus reduces the range of possible values for function f.

- *Closing the interval rule:* if the domain of function f is linear (ordered) and a selector conjunct is in the form [f(x)=a∨b] then the value set can be generalized to the closed interval [a,b] that is written in APC as [f(x)=a..b]. The composite value "a..b" replaces all values from a to b in the domain of f.

- *Climbing value hierarchy rule:* if the function f is annotated with a value hierarchy (or if one can be inferred from background knowledge) then a selector conjunct of the form [f(x)=a∨b∨c...] can have its value list generalized to a more general value g in the

hierarchy if values a,b,c,... are all special cases of value g. For example, in the value hierarchy *shape* shown in Fig. 4.2., the value *polygon* is a generalization of the values *triangle, square, rectangle.* The value *polygon* can replace a disjunction of two or more values that lie below it in the hierarchy.

Descriptor construction transformations (specializing transformations)

- *Counting quantified variables rule:* in an APC statement, quantified variables are annotated with a symbol denoting a quantified variable equivalence class. For example, in a blocks-world problem involving *boxes* and *objects* such as a situation described by

  $$\exists$$ box1,box2,object1, [ontop(box2,box1)][in(object1,box2)]

  some quantified variables belong to the variable class *boxes* (e.g., box1,box2) and others belong the the variable class *objects* (e.g., object1). The counting quantified variables rule constructs a selector that specifies the number of distinct quantified variables of each variable class present in a description. In the blocks-world situation above, the two selectors constructed are [#boxes=2] and [#objects=1], read "the number of boxes is 2"

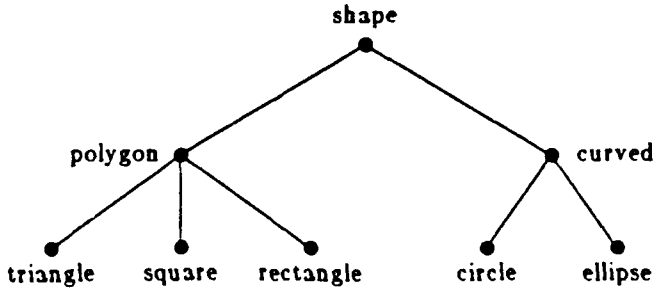


Figure 4.2. The value hierarchy for the descriptor *shape*.

and "the number of objects is 1." respectively. They are conjoined to the original APC expression to yield

∃ box1,box2,object1, [ontop(box2,box1)][in(object1,box2)][#boxes=2][#objects=1].

- *Counting similar selectors rule:* sometimes two or more selectors are conjoined and are identical except for quantified variable. For example (continuing with blocks-world situations) consider the statement

  ∃ box1,box2, [color(box1)=red][color(box2)=red].

  The counting similar selectors rule constructs a selector to indicate "there are two boxes that are red." This is written in APC as [#boxes(color=red)=2], with the general form

  [# <variable class> (<property>)= <count>].

- *Counting distinct values rule:* this rule constructs a selector for each function domain indicating the number of distinct values in the domain which appear as values in the APC statement. For example, in the statement

  ∃ box1,box2, [color(box1)=red][size(box1)=small][color(box2)=red][size(box2)=large]

  the constructed selectors that will be conjoined are

  [#diff-colors=1][#diff-sizes=2].

- *Universal properties rule:* a "forall" predicate is constructed when all functions of quantified variables from the same variable class appear with the same values. For example in the statement

  ∃ box1,box2,box3, [color(box1)=red][color(box2)=red][color(box3)=red]

  all boxes box1,box2,box3 have the property "color=red" which is denoted by conjoining the constructed selector [forall-boxes(color=red)]. Such selectors have the form

  [forall-<variable class> (<property>)] if true, or

  [forall-<variable class> (<property>) = FALSE] if false.

- *Generating chain properties rule:* two-place anti-symmetric[3] predicates (e.g., *ontop*) may form chains of relationships such as [ontop(A,B)][ontop(B,C)]. In such chains, there are extreme elements (denoted by two different quantified variables) at the ends of the chain. Let one quantified variable denote the "most-" end of the chain and let the other quantified variable denote the "least-" end. The chain properties rule constructs "most-" and "least-" predicates to denote the ends of the chain. For the above example, the additional selectors would be [MOST-ontop(A)] and [LEAST-ontop(C)].

- *Generating equivalence relations rule:* when two or more different quantified variables appear with the same function symbol and values, the entities are equivalent with respect to that function. Consider the statement

  $\exists$ box1,box2,box3, [color(box1)=red][color(box2)=red][color(box3)=blue].

  The quantified variables box1 and box2 have the same color function value (red). An equivalence selector is constructed as [SAME-color(box1.box2)]. The dot is used to delimit variables to denote symmetry (i.e., [SAME-color(box1.box2)] = [SAME-color(box2.box1)]).

- *Applying background inferences:* background knowledge is supplied in the form of APC inference rules (CONDITION $\Rightarrow$ CONSEQUENCE, where the consequence is a conjunction of selectors). Whenever a description satisfies the CONDITION part of a background rule, the CONSEQUENCE selectors are conjoined to the description (unless they are already present).

- *Using special computations, experiments or devices:* under special circumstances new descriptors can be obtained by performing certain computations (e.g., by the simulation of some process), by directing an automated experiment using computer-controlled apparatus,

3. The predicate ontop(A,B) is anti-symmetric if ontop(A,B) does not imply ontop(B,A). For constructing chain properties, it is the property of transitivity rather than anti-symmetry that is actually important. In the implementation of the "generating chain properties rule" the assumption is made that anti-symmetric predicates are also transitive. This assumption reflects the expressiveness of the APC language which can denote symmetric vs. anti-symmetric predicates but has no notation to differentiate transitive from intransitive ones. This approach can lead to potential (though minor) difficulties.

or by acquiring data directly from sensing devices. The implementations discussed in the thesis do not make use of these kinds of descriptor construction transformations.

4.4. Building classifications by repeated discrimination

The process of generating a hypothetical classification composed of k conceptual classes when given k seed objects is embodied in the algorithm CONCEPTUALIZE-SEEDS given in Fig. 4.3. The algorithm works in parallel on all k seeds by applying the inductive inference learning paradigm of *learning from examples* to k different positive example sets (each containing one unique seed object) and corresponding negative example sets containing the other k-1 objects.

Given are the examples (observed events) to be classified, the currently selected seed examples, the classification goal, and a search limit. The algorithm begins by generating k *stars* $G(e_i|F_i)$, i=1,2,...,k. Each such star is the set of all maximal-under-inclusion complexes which cover event e_i and none of the events in the *against* set F_i [Michalski, 1974]. The sets F_i are constructed to contain all seed events except e_i. Thus, the set $g_i=G(e_i|F_i)$ contains class descriptions that describe (cover) seed e_i and perhaps other events in set E but not any other seed.

This generates conceptual "localities" that are combined, made disjoint, and become hypothetical classifications, as described below. The classifications generated by considering different combinations of complexes are evaluated according to the classification goal and the best classification is retained. It is output when the search limit is reached.

A preliminary step of the classification building procedure is to remove any duplicate examples in the given collection of objects. Then each of the conceptual descriptions is generated in the following way. There is one positive example for the class being generated and k-1 negative examples of the class. Because there are no duplicate examples, it is known that the positive example is unlike any negative example. This assures that it is possible to build k-1 non-empty sets of generalized conjunctive descriptions that cover the positive example and do

Given: E, a collection of ovserved events,
k seeds $e_1, e_2, \ldots, e_k \in E$
H, the classification goal
L, the search limit

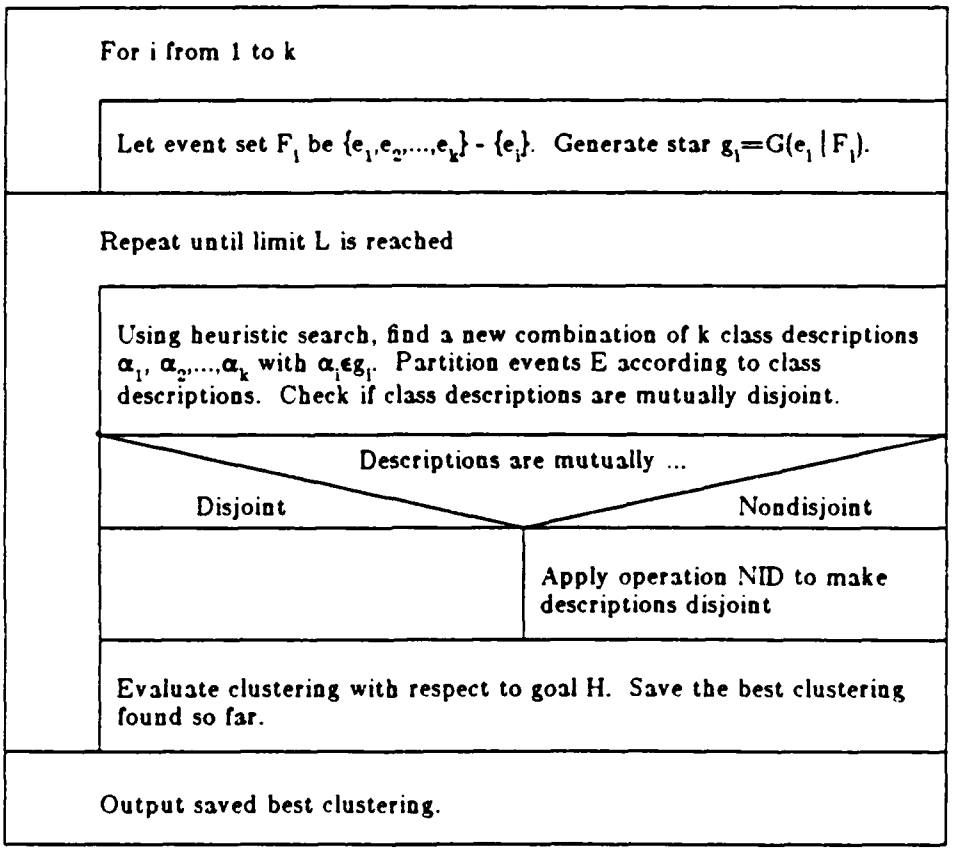


Figure 4.3. CONCEPTUALIZE-SEEDS algorithm.

t cover a unique negative example. When the k-1 conjunctive statements are logically ultiplied together, each hypothetical conjunctive description in the resulting set covers the sitive example and none of the negative examples. The classification goal evaluation criterion applied to the set of hypothetical descriptions and the best description is selected.

4.4.1. Selecting seeds

The seed events focus the inductive inference process and it is desirable that each seed event represent a strong pattern underlying the collection of events. Three selection procedures are utilized, under different situations.

In the first iteration, seed events are selected randomly or arbitrarily. Such a start is for convenience and reflects the typical situation in which nothing is known about patterns in the data *a priori*. Where background knowledge does contain such information, or where statistical tests or other conceptual data analysis tools are available, they can be used to indicate plausible representative seed events for the first iteration. In the absence of knowledge to be used to select the initial seed events, the first iteration of the algorithm (which uses background knowledge and the principle of conceptual cohesiveness to fit concepts to the seeds) supplies some knowledge for selecting seeds for the second iteration. Thus, the starting point of the first iteration is of minor importance and careful initial seed selection is not crucial for good results.

For each subsequent iteration of the algorithm, the last obtained result is either better than or worse than its predecessor. As long as the classifications generated are improving, concept focusing is enhanced by selecting from each class a seed which is most representative of that class. Such an event has the maximum number of characteristics in common with the other events in that class. For functions with values on a linear (ordinal) scale, the seed event should take a value for the function which is about average for all events in the class. The seed event is selected by summing the *syntactic distance* between it and all other events in the same class (syntactic distance is defined below).

When the last result is worse than the previous result, a *principle of adversity* is practiced. Under such a principle, the representative event is purposefully selected to cause "trouble," i.e., to severely test the conceptual cohesiveness of the class. This is done by selecting a seed event which is most unlike the others in the class (as measured by syntactic distance). When such a selection is made, two situations may develop. If the current classes reflect patterns of sufficient

strength, then any event within a class may be selected as the representative without affecting the class descriptions developed in the next iteration. Thus, in such a situation, the classification would not change. When the current hypothetical classification rests on patterns that are not strong, the selection of an extreme seed event will shift the classification to an entirely new partition of events and associated class descriptions. This heuristic operation encourages the algorithm to *discover* (explore) a new part of the problem solution space in the hope that some unexplored classification there will be better than all others previously generated.

The implementations of the algorithm include a mechanism that records each set of seeds as they are used in order to prevent the duplication of selected seeds (purely for the sake of efficiency). For some problems with just a few events, it is possible to exhaust all combinations of events such that no unused set of seed events can be constructed. In such cases, iteration halts before satisfying the regular stopping criterion.

The *syntactic distance* measure used to select seed events is a function $\delta(e_1,e_2)$ that is computed by summing the syntactic distances between the values of each variable in the events e_1 and e_2. The syntactic distance between two variable values is a number from 0 to 1, determined by a measure that reflects the domain type of the variable. For a nominal variable, the syntactic distance is either 0, if the values taken by the variable in each event are identical, or 1, if the values are not identical. For a linear variable, the syntactic distance is the ratio of the absolute difference between the values to the total span of the value set of the variable. For a structured variable, the evaluation of syntactic distance depends on the type of generalization hierarchy. Since structured variable values in events are leaves of a generalization hierarchy, the syntactic distance between such values for unordered and ordered hierarchies is evaluated the same way as for nominal and linear variables, respectively.

A note is appropriate to distinguish the syntactic distance measure from that of conceptual cohesiveness. Syntactic distance is a numerical function of two ℓ-complexes that sums (in the special way described above) differences in attribute values. It takes into account the knowledge

of the domain of the attribute, but not the fit of concepts to the events. It is a good heuristic for choosing an event which has attribute values that are mostly typical of the other events in the cluster. Syntactic distance is not a good basis for forming the clusters themselves. Conceptual cohesiveness is a good basis for forming clusters. Although the conceptual cohesiveness between two events is not evaluated in the mathematical sense, all events placed into the same class have maximal conceptual cohesiveness because they are described by the same concept. Likewise, events from different classes have minimal conceptual cohesiveness. Thus, in present implementations, conceptual cohesiveness is not a utility function, but is a built-in bias of the conceptual clustering algorithm.

4.4.2. Making concepts disjoint

One disadvantage of the presented method is that the class descriptions formed are not necessarily disjoint, though disjoint classifications are frequently desired. Within each class described by a non-disjoint class description there will be some events which are uniquely covered by the one class and other events which are multiply covered. The following algorithm called NID for transforming Non-disjoint descriptions Into Disjoint ones is used to adjust class membership just enough to make the classes and class descriptions disjoint.

Let set M be composed of all multiply covered events. By removing from all classes all events in set M, and then specializing the class descriptions to tightly cover the remaining events, a set of disjoint class descriptions will be generated. The next step will attempt to adjust class descriptions to again cover the uncovered events in set M.

Each event in set M is to be assigned to a class, if possible. One event e in set M is fitted to a class by building a class description that covers all events permanently assigned to the class and event e, for each class. Only one of the k possible class assignments for event e will be retained, determined by how well each augmented class meets the classification goal criterion. Once the best class for event e is determined, the event becomes a permanent member of that

class.

4.4.3. Applying a stopping criterion

As the iterative algorithm progresses, there is a tendency (introduced by the seed selection technique) for the classification to improve. Improvement is not monotonic, however, and because heuristics are used in virtually every operation to combat the combinatorial explosion that would otherwise occur, each part of the algorithm can only proceed towards a quasi-optimal result.

Because each iteration can potentially lead to a new "best" result (with respect to the classification goal), the temptation is to iterate forever. A simple formula is used to balance the expectation of improved results vs. the computational time consumed. The formula includes the desire to produce not just one single result but several alternative results (remember that there is no "right" answer in classification building). The stopping criterion is based on two user-supplied numbers called *base* and *probe*. *Base* denotes the number of alternative results desired. This number of iterations are necessary as a minimum to ensure that sufficient alternative results exist to report. After *base* number of iterations, *probe* number of additional iterations are performed. *Probe* indicates the width (in terms of iterations) of "voids" between local optima which can be bridged. If *probe* number of iterations are performed without finding a result better than any of the *base* number of saved alternative results, the stopping criterion is met and the algorithm halts. Each time a better classification is generated, it is added to the alternative classifications list after the worst one in the list is discarded and the probe counter resets to count another *probe* number of iterations. The net effect is that the algorithm continues towards the optimum result until the iteration spacing between the next local optimum and the previous one exceeds probe iterations.

CHAPTER 5

METHOD 1: CLUSTERING UNSTRUCTURED OBJECTS

5.1. Overall approach

This chapter presents the conjunctive conceptual clustering methodology sufficient to operate on unstructured objects. The input consists of a collection of examples (and optional negative examples) each described as an event in the form of a complex which is a vector of attribute values (i.e., zero-place functions and their values). Every event has a value for every attribute. The output consists of a classification which is a hierarchy of classes in which each node in the hierarchy is described by a quantifier-free APC conjunctive statement.[1] In the class hierarchy, all internal nodes branch into at most k_{max} disjoint subclasses.

Background knowledge is a secondary input to the conceptual clustering process. In this chapter, two kinds of background knowledge are considered. The first kind is provided by the APC variable annotations. It consists of the domain type (nominal, linearly ordered, hierarchically structured), the number of values in the domain, an optional cost of the variable, and (for structured domains) the generalization hierarchy of values. The second kind of background knowledge consists of optional APC implicative statements, as described in Sec. 1.2. In the method described in this chapter, the implicative statements use only zero-place functions and predicates and are quantifier-free.

The algorithm consists of a *clustering module* and a *hierarchy-building module*, which are described in Secs. 5.3 and 5.5, respectively. The clustering module is logically decomposed into four main subalgorithms which are described for clarity in the following section prior to the

1 This subset of APC is the Variable-valued Logic system 1 (VL$_1$; Michalski 1974

description of whole clustering module.

5.2. Main subalgorithms of the clustering module

5.2.1. Criterion evaluation

The problem of how to judge the quality of a classification produced by conceptual clustering is difficult, and there seems to be no universal answer to it. One can, however, indicate two major criteria. The first is that descriptions formulated for classes should be *simple*, i.e., that it be easy to assign objects to classes and to differentiate between the classes. This criterion alone, however, could lead to trivial and arbitrary classifications. The second criterion is that class descriptions should *fit well* the actual data. To achieve a very precise fit, however, a description may have to be complicated. Consequently, the demands for simplicity and good fit are conflicting, and the solution is to find a balance between the two.

A number of other measures can be introduced for evaluating clustering quality. One can use a combined measure that includes any of the following elementary criteria:

- The *fit* between a classification and the data is computed in two different ways, denoted as T and P. The T measure is the negative of the Total sparseness of the clustering, and the P measure is the negative of the sum of the Projected sparsenesses of the complexes. (The reason for using the negative values is to increase the degree of match as the sparseness decreases.) The *sparseness* of a complex is the number of unobserved events it covers.[2] Given some events E covered by complexes α and α', the complex with the high sparseness is the more general. When good fit is desired, the sparseness should be low. The Total sparseness is calculated when all dimensions of the event space are considered when locating and counting the unobserved events. The Projected sparseness is determined when only certain dimensions corresponding to attributes which take different values in

2 When events are weighted, the sparseness is computed as the number of unobserved events plus the number of observed events minus the sum of the weights of the observed events. When using weighted events it is possible for the sparseness to be negative

two or more class descriptions are used to locate and count the unobserved events.

- The *simplicity* of a clustering is defined as the negative of its complexity, which is the sum of costs attrib ted to each selector present in the complexes. The selector cost function is based on the number of elements found in the selector's value list and a user-defined cost associated with the function/predicate symbol. Selectors with few values are less complex than those with many values, and therefore should have a smaller cost. The cost function for selector i is calculated as $(NVL_i\text{-}1) + COST_i$, where NVL_i is the number of symbols in the value list of the selector and $COST_i$ is the cost associated with the function/predicate symbol used in the selector. If not specified by the user, $COST_i$ takes a default value of 1.

- The *commonality* of a class description is the number of properties (i.e., attribute/value pairs) shared by the events in the class. The commonality is measured by counting the number of selectors that appear in the class description. This criterion is analogous to the traditional clustering criterion of maximizing an event-to-event similarity score (e.g., number of shared properties) for events within the same cluster.

- *Inter-cluster difference* is measured by the sum of the degrees of disjointness between every pair of complexes in the clustering. The degree of disjointness of a pair of complexes is the number of selectors in both complexes after removing selectors that intersect or that have no counterpart containing the same variable symbol. For example, the pair of complexes:

  [*color=red*][size=small∨medium][shape=circle]

  [*color=blue*][size=medium∨large]

  has the degree of disjointness 2, because 2 of the 5 selectors intersect and the one with the variable *shape* has no counterpart. The disjoint selectors are italicized. This criterion promotes clusterings with classes having many differing properties, and is analogous to the criterion requiring maximal distance between clusters used in conventional methods of clustering.

- The *discrimination index* is the number of variables that singly discriminate among all the classes, i.e., variables having different values in every class description.

- *Dimensionality reduction* is measured by the difference between the total number of variables and the essential dimensionality defined as the minimum number of variables required to distinguish among all complexes in a clustering. It can be computed by applying the variable-valued logic minimization algorithm A^q [Michalski, 1972] to the clustering. When the discrimination index is greater than zero (i.e., when there exists a variable which singly discriminates all classes) and there are n variables, the dimensionality reduction achieves its maximum value of n-1.

The definitions of the above criteria are such that the increase of any criterion value improves the quality of the clustering. The relative influence of each criterion is specified using the *Lexicographical Evaluation Functional with tolerances* (LEF) [Michalski, 1980a]. The LEF is defined by a sequence of "criterion-tolerance" pairs (c_1,τ_1), (c_2,τ_2), ..., where c_i is an elementary criterion selected from the above list, and τ_i is a "tolerance threshold" ($\tau \in [0...100\%]$). In the first step, all clusterings are evaluated on the first criterion, c_1, and those that score best or within the range defined by the threshold τ_1 are retained. Next, the retained clusterings are evaluated on criterion c_2 with threshold τ_2, similarly to the above. This process continues until either the set of retained clusterings is reduced to a singleton (the "best" clustering) or the sequence of criterion-tolerance pairs is exhausted. In the latter case, the retained clusterings have equivalent quality with respect to the given LEF, and any one may be chosen arbitrarily. The selection of elementary criteria, their ordering, and the specification of tolerances is made by a data analyst by constructing a LEF.

5.2.2. Generating a conceptual cover of events

A description covering two or more events and/or class descriptions (complexes) is generated by the REFUNION operation. For each variable, the set of all values the variable

takes, in all given events and complexes, is determined. These sets are used as the reference values of the variable in the generated complex. For example, given complexes describing two events e_1 and e_2 and another complex α:

$$e_1 : [x_1=2][x_2=3][x_3=0][x_4=1]$$

$$e_2 : [x_1=0][x_2=2][x_3=1][x_4=1]$$

$$\alpha : [x_1=2..3][x_2=4][x_3=0][x_4=2]$$

the refunion complex REFUNION(e_1,e_2,α) that covers e_1, e_2, and α is

$$[x_1=0 \vee 2 \vee 3][x_2=2 \vee 3 \vee 4][x_3=0 \vee 1][x_4=1 \vee 2].$$

A refunion-generated complex has minimum sparseness (both absolute and projected) among all complexes covering all the given entities.

5.2.3. Generalization transformations

All of the generalizing transformations introduced in Chapter 4 can be applied to unstructured event descriptions. This includes the *dropping condition rule*, the *adding alternative value rule*, the *closing the interval rule*, and the *climbing value hierarchy rule*. Two descriptor construction transformations can be applied in the absence of quantified variables. They are the *counting distinct references rule*, and the *applying background inferences rule*.

The descriptor construction transformations are applied to event descriptions, as described in Chapter 4. The generalizing transformations are applied by procedure GENRLIZ. For simplicity, most description-generating operations (e.g, REFUNION, REDUSTAR, etc.) are carried out initially as if all variable domains were nominal (i.e., not subjected to such generalizations as *closing the interval* and *climbing the value hierarchy*). Then the GENRLIZ procedure applies all generalization transformations at once and appropriately adjusts reference sets in the selectors.

5.2.4. Building a star

The *star* G(e|F) *of event* e *against event set* F ($e \notin F$) is the set of all maximally general[3] complexes covering the event e and not covering any event in F. Informally, it is the set of all maximally general descriptions of event e which do not intersect with set F. Figure 5.1 presents a star of event e against events denoted by "•" in the two-dimensional space spanned over linear variables. The star consists of complexes α_1, α_2, and α_3. Complex α'_3 is a "reduced" complex α_3, as explained below.

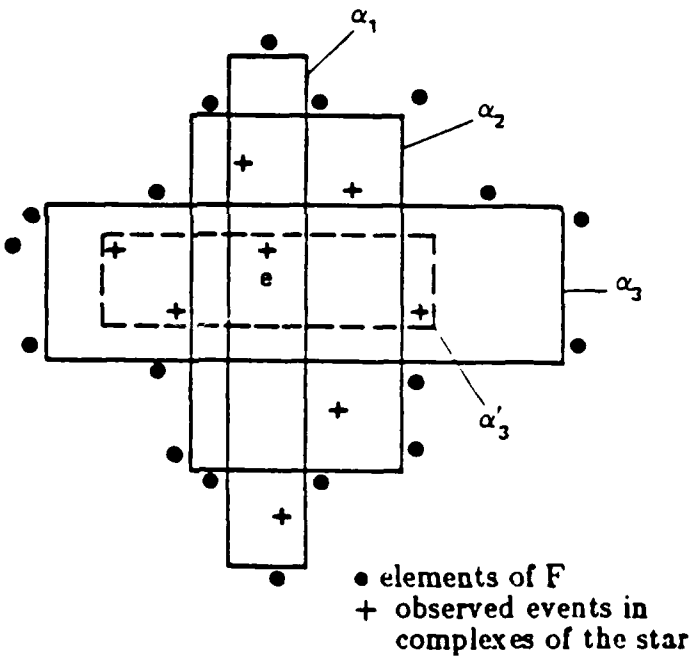


Figure 5.1. An illustration of the star G(e|F).

3. A complex α is maximally general with respect to a property P if there does not exist a complex α^* with property P such that $\alpha \subset \alpha^*$

In the algorithm described in the next section, the "theoretical" stars (defined in the above paragraph) are subjected to two major modifications. The first is to minimize the sparseness of complexes in the stars, and the second is to "bound" the stars, i.e., to select from them a certain number of "best" complexes, according to a context-dependent criterion. The first modification is performed by algorithm REDUSTAR, described below, and the second by algorithm BOUNDSTAR described in Sec. 5.4.

Complexes in stars G(e|F) are maximally general, and therefore may describe objects in an overgeneralized way. The algorithm REDUSTAR generates a star and then maximally reduces the sparseness of each complex in it, while preserving the coverage of observed events. For example, complex α'_3 in Figure 5.1 is such a reduced complex obtained from complex α_3. The steps of the REDUSTAR procedure are:

(1) Elementary stars, $G(e|e_i)$, $e_i \in F$, are determined

To generate an elementary star $G(e|e_i)$ of an event e against another event e_i, all variables that have different values in e than in e_i are identified. Suppose, with no loss of generality, that these variables are $x_1, x_2, \ldots, x_g$, and that $e_i = [x_1=r_1][x_2=r_2]\ldots[x_g=r_g]\ldots[x_n=r_n]$. The complexes of the star $G(e|e_i)$ are then $[x_j \neq r_j]$, $j=1,2,\ldots,g$, because these are the maximally general complexes that cover e and do not cover e_i. The number of complexes in an elementary star is at most n, and, because $e_i \neq e$, at least 1.

(2) The complete star G(e|F) is determined

The star G(e|F) is generated by first setting up the logical product $\bigwedge G^\circ(e|e_i)$, $e_i \in F$, where $G^\circ(e|e_i)$ is the disjunction of complexes from the elementary star $G(e|e_i)$ [Michalski, 1974]. The multiplication of complexes is performed, using absorption laws, until a disjunction of irredundant complexes is obtained. This multiplication is carried out in steps, each step being a multiplication of a disjunction of complexes by a disjunction of selectors (the elements of

consecutive elementary stars). The set of the complexes in the resulting disjunction is G(e|F).

(3) Complexes in G(e|F) are reduced and simplified

The sparseness of each complex in the star is reduced as much as possible without "uncovering" any of the observed events. This is done by performing the REFUNION of all the observed events contained in each complex. The obtained complexes are then generalized and simplified by applying the GENRLIZ operator.

5.2.5. NID: Making nondisjoint complexes disjoint

This procedure transforms a set of Nondisjoint complexes Into a set of Disjoint complexes (i.e., a disjoint clustering). If input complexes to NID are already disjoint, the procedure leaves them unchanged. The steps of the procedure are:

(1) 'Core' complexes are determined

Observed events covered by more than one complex from the given set are placed on the *multiply-covered event list* (m-list). If the m-list is empty, then the complexes are only weakly intersecting, i.e., the intersection area contains only unobserved events. In this case, the procedure terminates with an indication that the combination of complexes is a *weakly intersecting clustering*. Otherwise, each complex is replaced by the REFUNION of the observed events contained in the complex that are not on the m-list (i.e., that are singly covered). The obtained refunions are called "core" complexes.

(2) A best 'host' complex is determined for each event on the m-list

An event is selected from the m-list and is "added" to each of the k core complexes by generalizing each complex to the extent necessary to cover the event. Such a generalization is performed by applying the REFUNION operator to the event and the complex. As a result, k modified complexes are obtained. By replacing one of the core complexes in the initial set with the corresponding modified complex, in k different ways, a collection of clusterings is obtained.

These clusterings are evaluated according to the given clustering quality criterion. The complex in the best clustering that covers the given event from the m-list is considered to be the best "host" for this event. The best clustering is retained and the remaining ones are eliminated. By repeating the above operation for every event on the m-list, a set of k disjoint complexes is obtained whose union covers the same observed events as the original set of nondisjoint complexes.

If an event cannot be "added" to any complex without causing the result to intersect other complexes, then the event is placed on the *exceptions list*.

5.3. The clustering module

The fundamental subalgorithms described above are used as component parts to build a composite algorithm (module) for generating a conjunctive conceptual clustering that splits the collection of events into a given number of classes. The general algorithm underlying the implementation of the clustering module was introduced by Michalski [1980b] and described in Chapter 4. Its goal can be described as follows:

Given: E, A collection of events to be clustered

k, The number of clusters desired

H, The goal or criterion of clustering quality LEF

F′, An optional set of negative events

Find: A disjoint clustering of the collection of events E that optimizes the given criterion of clustering quality H and covers no events in set F′.

A straight forward, exhaustive-search based version of the algorithm is described first and then modifications are presented to increase its efficiency. The algorithm is summarized by the flow

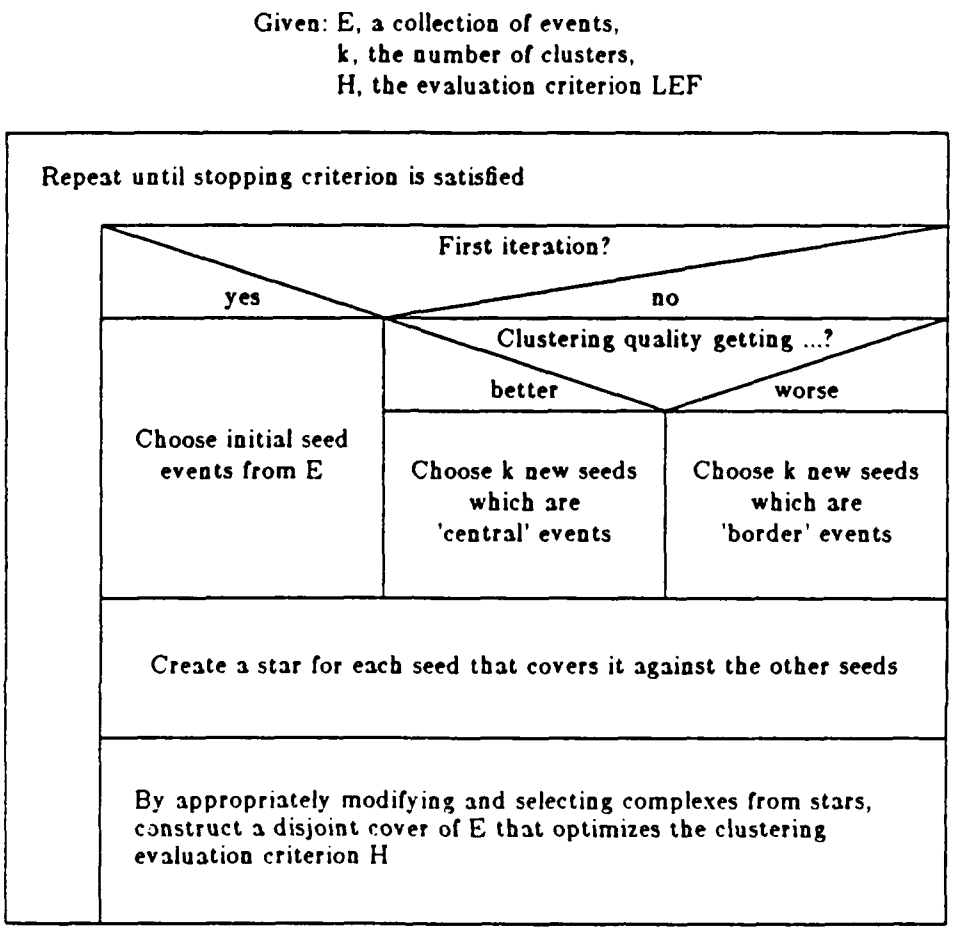


Figure 5.2. The flow chart of the clustering module.

chart in Fig. 5.2. The steps of the algorithm are:

(1) Determine initial seeds for first iteration

From the given collection of events E, k events (the *initial seeds*) are selected. The seeds may be chosen randomly or according to some criterion. (After this first step, seeds are always

selected according to certain rules; see step 5).

(2) Construct stars for seeds

For each seed e_i, a reduced star $RG(e_i|F)$ is constructed by algorithm REDUSTAR, where F is the set of remaining seeds union F′ (F′ is often empty).

(3) Construct a disjoint cover by selecting and modifying complexes from stars

Every combination of complexes, created by selecting one complex from each star, is tested to see whether it contains intersecting complexes. If so, the complexes are modified by procedure NID to make them disjoint. An optimized clustering that is a disjoint cover of E is produced.

(4) Evaluate stopping criterion

If this is the first iteration, the obtained clustering is stored. In subsequent iterations the clustering is stored only if it scores better than previously stored clusterings according to the LEF. The algorithm terminates when a specified number of iterations does not produce a better clustering, according to the *stopping criterion* described in Sec. 4.3.3).

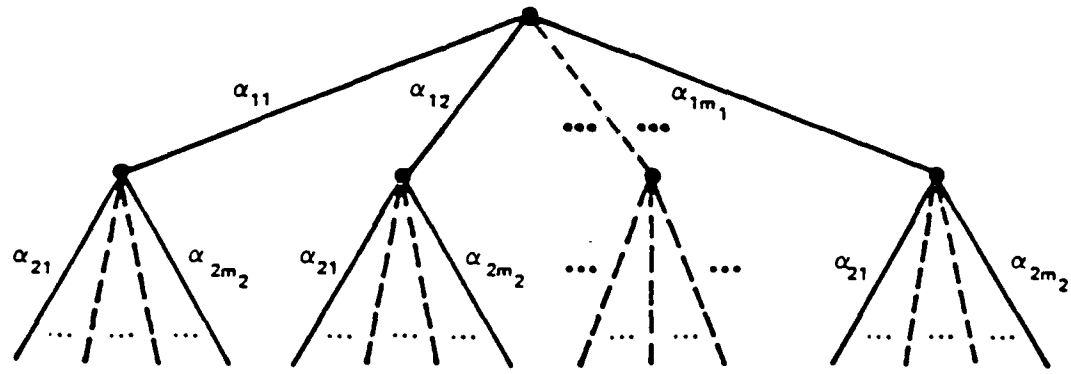
(5) Select new seeds

New seeds are selected from sets of observed events contained in complexes of the generated clustering, one seed per complex. Two seed selection techniques are used. One technique selects "central" events, defined as events nearest the geometrical centers of the complexes (as determined by the syntactic distance). The other technique, stemming from the "adversity principle,"[4] selects "border" events, defined as events farthest from the centers. Ties for central or border events are broken in favor of events that have not been used recently as seeds. The technique of selecting central events is used repetitively in consecutive iterations as

4 This principle states that if a border, "near hit" event truly belongs to the given cluster, then when selected as the seed it should produce a clustering that contains the same events as when a central event is used as a seed

long as the clusterings improve. When the improvement ceases, border events are selected. After selecting new seeds, the algorithm continues from step 2. Along with a clustering, the algorithm generates k class descriptions (complexes) describing individual clusters, and determines how well these complexes score on the evaluation criteria in the LEF.

The most computationally costly part of this algorithm is the construction of an optimized clustering, given k seed events (step 3). For an illustration, let us assume that k=2 and that k "seeds", e_1 and e_2, have been selected from the collection E. In the first step, star G_1 is G(e_1|remaining seeds) and star G_2 is G(e_2|remaining seeds). Fig. 5.3 presents complexes of these stars as branches of a search tree. Branches from the root represent complexes of star G_1 that are α_{11}, α_{12}, ..., α_{1m_1}, and branches at the second level (repeated m_1 times) represent complexes of star G_2 that are α_{21}, α_{22}, ..., α_{2m_2}. Each combination of complexes, containing one complex from each star, corresponds to one path in the tree. Because any such combination may contain



α_{ij} denotes a complex j from star i.

Figure 5.3. The exhaustive search tree for k=2.

intersecting complexes, procedure NID is applied to each, and the result is a disjoint clustering These clusterings are ordered according to the quality criterion LEF, and the best one is selected.

5.4. Path-rank-ordered search

The above strategy for determining a clustering from seeds is very simple, but unfortunately too inefficient for solving any interesting practical problems. This is due to the fact that the stars may contain very many complexes. When there are n variables and k seeds, a star may contain up to n^{k-1} complexes (there are at most n complexes in any of the k-1 elementary stars needed to compute the complete star). Thus, when n=30 and k=3, there could be up to n^{k-1}=900 complexes, and the search tree could have up to 900-way branching at each node, and up to 900^3=729 million leaves. Absorption laws (as defined in set theory) will usually eliminate many redundant complexes, but the star may still be too large. Artificial intelligence research on various heuristic search procedures offers various possibilities for reducing the search [e.g., Nilsson, 1980; Winston, 1977]. To solve this problem, some of the known search strategies have been incorporated with some newly developed ones. The result is a search procedure called Path-Rank-Ordered (PRO) search that incorporates the following four techniques:

(1) Bounding the stars (procedure BOUNDSTAR)

The number of complexes in a star is bounded by a fixed integer m, which assures that the search tree has at most m-way branching. A bounded star contains not just m arbitrary complexes from the initial star, but the m "best" ones.

At each step of star generation (a multiplication of a set of complexes by the next elementary star, see STAR algorithm in Sec. 3), complexes are first reduced and then arranged in descending order according to the assumed clustering quality criterion LEF. Only the first m

complexes are retained for the next multiplication step. This operation is also performed at the end of star generation, so that the final star has at most m complexes. The stars so obtained are called *bounded reduced stars* and denoted G(e|F,m)

Some elementary criteria measure global properties of a clustering rather than properties of just a single complex (e.g., the inter-cluster differences). Consequently, when evaluating a complex descending from a node in the search tree that is not the root, the complex is evaluated in the context of complexes associated with the path from the root to this node.

By bounding the star, the process gains significantly in efficiency but gives up the assurance that the obtained clustering will be optimal. This is not a significant loss, however, because the clustering obtained at the end of each iteration contributes only the seeds to the next iteration, and thus its precise expression is not very important.

(2) Generating stars dynamically

Because it is necessary to evaluate complexes in the context of previously selected complexes, bounding a star has to be done differently at each node of the search tree. A "lazy" star generation strategy is used, in which a star is generated only when it is needed to expand a node on the path being explored.

(3) Searching in order of path rank

As mentioned above, complexes in a bounded star are arranged in descending order according to the LEF. In the search tree, the branch to the best complex is assigned the *branch index* 0, the branch to the next best complex is assigned the branch index 1, etc., up to the index m-1. The *path index* of a path from the root to a leaf is the sum of the branch indices along the path.

The paths from the root to a leaf represent potential clusterings and are investigated in the ascending order of their path index. Thus, the first path investigated is the one with path index

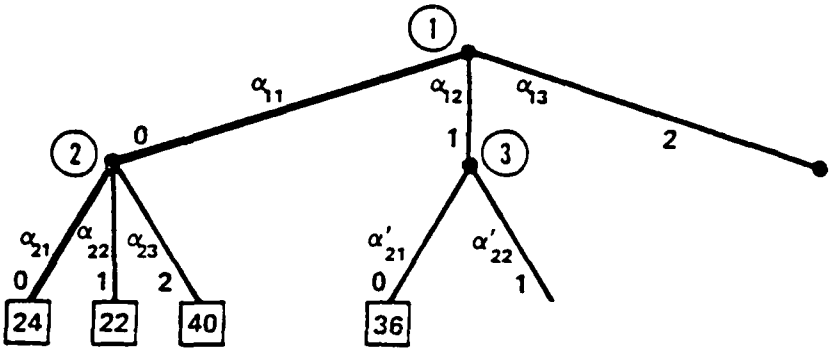
0, i.e., the path containing only the best complexes from each star. The next paths considered are those with a path index of one. There are k such paths.

As paths of increasing path index are generated and evaluated, a search termination criterion is applied. This criterion consists of two parameters, *search-base* and *search-probe*. A search-base number of paths is always expanded and evaluated. Then, a search-probe number of additional paths is considered. Each path is processed by NID, and if some complexes are transformed to make them disjoint, the clustering quality criterion is evaluated again. Whenever a new clustering is better than any previous clustering, it is saved and another search-probe number of additional paths is explored. If the above probing fails to find a better clustering, the search terminates.

(4) Tapering the search tree

The bound of the stars, m, is decreased with the increase of the path index. The search tree is, therefore, more fully developed on the side containing the "higher quality" complexes.

Fig. 5.4 shows an example of a search tree generated by path rank ordered search. The tree is a modification of the tree in Fig. 5.3, resulting from the application of the above efficiency-increasing techniques. In Fig. 5.4, the maximum bound m_{max} is set to 3. The root is expanded by constructing the star G(seed$_1$|other seeds,3), whose complexes are α_{11}, α_{12}, and α_{13} (listed in decreasing order of their "quality," as determined by the LEF). The branches representing these complexes are assigned branch indices 0, 1, and 2, respectively. The node attached to branch 0 is expanded next. The star G(seed$_2$|other seeds,3) is generated, creating complexes α_{21}, α_{22}, and α_{23}. Branches corresponding to these complexes are assigned branch indices 0, 1, and 2, respectively. The path 0-0 (having the lowest path index of 0, denoted by heavy lines in Fig. 5.4) is considered first. The associated clustering $\{\alpha_{11}, \alpha_{21}\}$ is processed by NID, and the result is saved as the best clustering so far. Next, path 0-1 is considered. The associated clustering $\{\alpha_{11}, \alpha_{22}\}$ is processed by NID and evaluated. If it is better than the



α_{ij} denotes complex j from star i. Integers (1), (2), ... indicate the order of expanding nodes. Integers 0, 1, ... indicate the branch indices. Integers [24], [22], ... indicate clustering evaluation scores for each path.

Figure 5.4. The path-rank-ordered (PRO) search tree for k=2.

previous clustering, it is saved. In order to explore the path 1-0 (the second path with path index 1), the star G(seed$_2$ | other seeds,2) is generated. It contains complexes α'_{21} and α'_{22}. The clustering $\{\alpha_{12}', \alpha'_{21}\}$ associated with this path is evaluated. Assuming that the termination criterion has the parameters search-base=2 and search-probe=2, and that the evaluation scores are as shown in Fig. 5.4, the tree search terminates after investigating the fourth path 0-2 (since this path exhausts the probing without finding a better clustering). Path 0-1, with the evaluation score of 22, is the best clustering found.

5.5. The hierarchy building module

The hierarchy-building module uses the clustering module to create a hierarchy of clusters. It performs two loops, one iterative and one recursive. The iterative loop repeats the clustering module for a sequence of values of k in order to determine the value for which the most desirable clustering is obtained. Such an approach is computationally acceptable because, in

practical applications, most interesting hierarchies will have a relatively small number of branches (i.e., value of k) at each level.

The recursive loop applies the above iterative process at each node of the hierarchy. In the first step, the process is executed for the root, representing the initial event set E. Clusters of E and their conjunctive descriptions are determined. Consecutive steps repeat the same operation for the nodes representing clusters obtained during the previous step. The hierarchy continues to grow from the top down until the *continue-growth* condition fails to be met. This condition is composed of three parts: the height of the hierarchy must be below the height limit MAXHEIGHT, the number of events in a cluster must be above a minimum number MINSIZE, and the fit between the clusters and their descriptions at the current level of the hierarchy must be better than at the previous level.

In order to determine the optimal value of k at each node, the clustering quality criterion must be modified so that it can compare clusterings with different numbers of complexes. Such a criterion must reflect the dependency of the fit between the clustering and data on the value of k. As the number of clusters k increases, the fit (measured by the negative of sparseness) will likely increase, since smaller complexes will have smaller sparseness. On the other hand, increasing k increases the complexity of the clustering and therefore is undesirable. A simple criterion that takes into consideration the above trade-off is to require the product

$$\text{Total sparseness} \times k^{\beta}$$

to achieve minimum value, where β is an experimentally determined parameter balancing the relative effect of the sparseness and the number of clusters k on the solution.

5.6. Dynamic modification of classifications

The classification obtained by the above procedures partitions all observed events into disjoint conceptual classes. Note that the set-theoretic union of all classes in the clustering does not necessarily cover the whole event space, i.e., there may be unobserved events which are not

covered by any class description.

In some problem situations it may be desired to revise the classification incrementally to cover new events as they are added to the collection of observed events. This can be done using the procedure NID as follows. The complexes of the current clustering play the role of "core" complexes, and the new event is treated as an element of the m-list. The event is incorporated into the complex that is the best "host" for it, as determined by the classification goal criterion. During this procedure, an original complex is replaced by the REFUNION of itself and the new event. NID performs this adjustment on each class description, keeping the one that best meets the classification goal, and returning the other descriptions to their original state.

Such a process has a psychological justification, as it is common for people to modify their classifications when some object fails to fit them, by appropriately perturbing the boundaries of the classes.

5.7. An example problem: Rediscovering soybean diseases

The problem is to reconstruct a classification of selected soybean diseases. Given are 47 examples of soybean diseases each characterized by the 35 multi-valued variables shown in Fig. 5.5. The input data for this problem is given in Appendix A along with output solutions as generated by the program. These examples are drawn from 4 populations—each population representing one of the following soybean diseases: Diaporthe stem canker (D1), Charcoal rot (D2), Rhizoctonia root rot (D3), and Phytophthora rot (D4).

Ideally, a clustering method should partition these given cases into four groups corresponding to the diseases. To test this, the unstructured classification building program (CLUSTER/2 [Stepp, 1984]) was applied to this problem. To contrast the conceptual clustering results with those of traditional numerical taxonomy, the reader is asked to return briefly to Fig. 2.1 which shows a typical dendrogram produced by a numerical taxonomy program from the same soybean data. The dendrogram is cut at the wavy line to create four classes. Classes 1, 3,

Time of occurrence	(7)	Condition of stem	(2)
Plant stand	(2)	Presence of lodging	(2)
Precipitation	(3)	Stem cankers	(4)
Temperature	(3)	Canker lesion color	(4)
Occurrence of hail	(2)	Fruiting bodies on stem	(2)
Number of years crop repeated	(4)	External decay of stem	(3)
Damaged area	(4)	Mycelium on stem	(2)
Severity	(3)	Internal discoloration	(3)
Seed treatment	(3)	Sclerotia	(2)
Seed germination	(3)	Condition of fruit pods	(4)
Plant height	(2)	Fruit spots	(5)
Condition of leaves	(2)	Condition of seed	(2)
Leaf spots—halos	(3)	Seed mold growth	(2)
Leaf spots—margin	(3)	Seed discoloration	(2)
Leaf spot size	(3)	Seed size	(2)
Shotholing/shredding	(2)	Seed shriveling	(2)
Leaf malformation	(2)	Condition of roots	(3)
Leaf mildew growth	(3)		

(Integers in parenthesis indicate the number of values in the domains.)

Figure 5.5. Variables used to describe diseased soybeans.

and 4 contain events from the same disease category (coded "D1", "D3", and "D2", respectively). Class 2, however, contains cases of both diseases D3 and D4.

The program CLUSTER/2 was applied to the problem with "maximizing the fit" as the evaluation criterion (LEF). The program partitioned the disease cases into four disease categories and described the clusters in terms of the characteristics (patterns of symptoms) of each disease, expressed in the form of a conjunctive statement. The produced disease categories corresponded exactly to actual soybean diseases, and the descriptions produced by the program corresponded to the symptoms indicated by plant pathologists for these diseases.

Figure 5.6 presents the complete description for one class (one disease category). The middle column contains the values for the 25 variables the program used to describe the class. The right-hand column of Figure 5.6 presents values of variables used by an expert plant

Variable	Value range determined by CLUSTER/2	Value range determined by plant pathologist
Time of occurrence	July to October	August to September
Precipitation	above normal	normal or above normal
Temperature	normal	normal or above normal
No. yrs. crop repeated	more than one	more than one
Stem cankers	above second node	above second node
Canker lesion color	brown or n.a.	brown
Fruiting bodies	present	present
Condition of fruit pods	normal	normal
Plant stand	normal	not present in expert descriptions
Damaged area	scattered areas or low areas	
Severity	potentially severe or severe	
Seed treatment	none or fungicide	
Plant height	abnormal	
Condition of leaves	abnormal	
Leaf spots—halos	absent	
Leaf spots—margin	irrelevant	
Shotholing/shreading	absent	
Leaf malformation	absent	
Leaf mildew growth	absent	
Condition of stem	abnormal	
External decay of stem	firm and dry	
Mycelium on stem	absent	
Int. discolor. of stem	none	
Sclerotia int. or ext.	absent	
Condition of fruit pods	normal	
Fruit spots	irrelevant	
Condition of seed	normal	
Seed mold growth	absent	
Seed discoloration	absent	
Seed size	normal	
Seed shriveling	absent	
Condition of roots	normal	

Figure 5.6. The generated description of one class (Diaporthe Stem Canker).

pathologist to describe the same disease for diagnosis. The description of the disease determined by the program contains all the symptoms of the disease specified by the plant pathologist. The values of "Time of occurrence," "Precipitation," and "Canker lesion color" determined by the program are supersets of the values mentioned by the plant pathologist. The slightly expanded value ranges for these attributes determined by the program are correct in the sense that disease

Variable	Class D1 (Diaporthe stem canker)	Class D2 (Charcoal rot)	Class D3 (Rhizoctonia root rot)	Class D4 (Phytophthora rot)
Plant stand	normal	normal	irrelevant	less than normal
Precipitation	above normal	below normal	above normal	normal or above
Temperature	normal	normal or above	below normal	normal or below
Damaged areas	scattered or low areas	whole fields, upland areas	low areas	whole fields, low areas
Stem cankers	above 2nd node	absent	below soil line	below or slightly above soil
Canker lesion color	brown	tan	brown	dark br. or black
Fruiting bodies on stem	present	absent	absent	absent
External decay of stem	firm and dry	absent	firm and dry	absent or firm and dry
Int. discolor. of stem	none	black	none	none
Sclerotia int. or ext.	absent	present	absent	absent
Condition of fruit pods	normal	normal	irrelevant	irrelevant
Condition of roots	normal	normal	normal or rotted	rotted

Figure 5.7. A summary of discriminant variables in the generated classification.

cases exhibiting values that lie outside the ranges mentioned by the pathologist are classified by human experts as cases withing the disease category identified by the program.

The generated descriptions also involve many variables which the plant pathologist did not mention. Fig. 5.7 shows a table of the values of the discriminant variables for each of the four classes found in the top level of the classification hierarchy generated by CLUSTER/2. The entire two-level hierarchy is shown in Fig. 5.8 with the links between subclasses labeled with the minimal set of discriminant variables selected from the complete descriptions of each subclass.

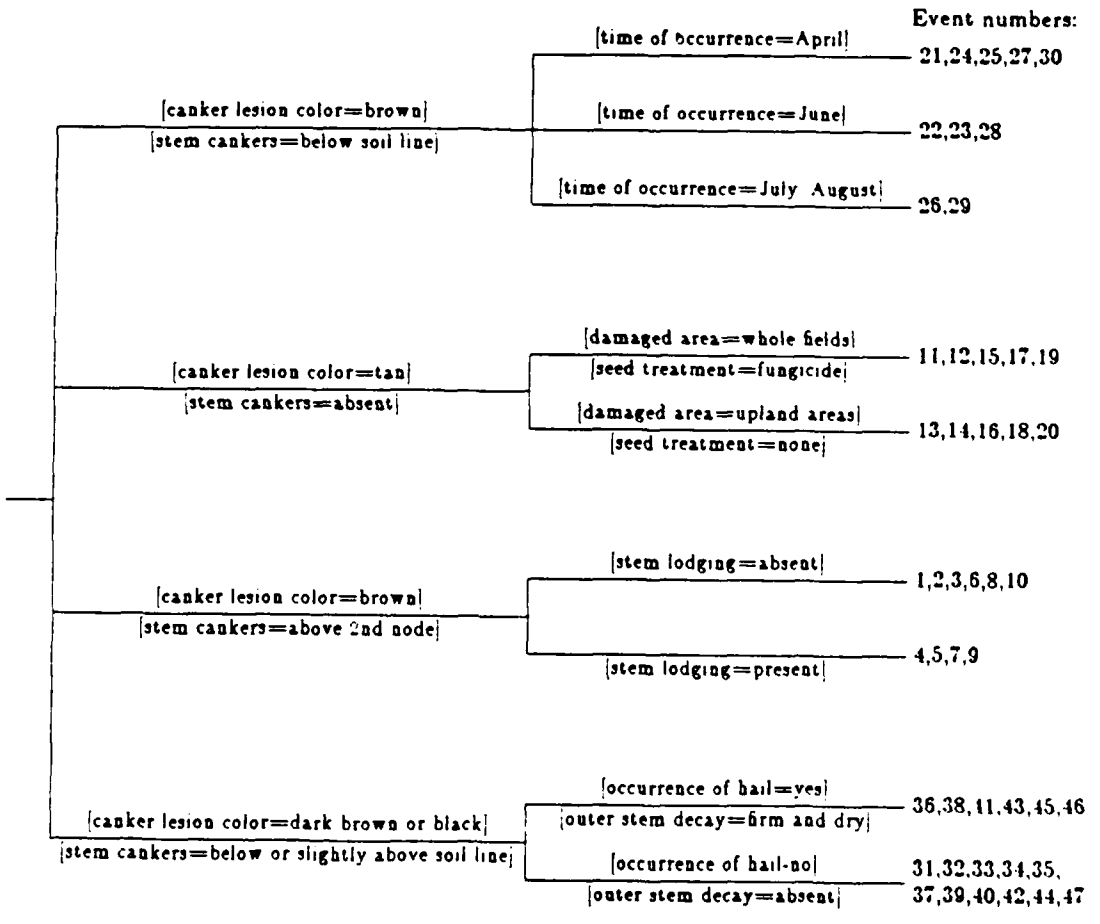


Figure 5.8. A two-level classification of diseased soybeans.

5.8. Other applications of attribute-based clustering

The presented method of attribute-based conceptual clustering is domain independent and this has encouraged its application to a wide variety of problems. In one experiment, the method was used to build a classification of patients with various craniosynostosis syndromes (e.g., Crouzon, Saethre-Chotzen, and Apert syndromes) [Michalski, Baskin, Spackman, 1982]. The data consisted of 18 binary attributes indicating the presence or absence of physiological features, such as craniostenosis, facial asymmetry, cleft palate, spinal malformation, etc. Using the goal of achieving good fit of the classification to the data, two classifications of 14 patients were made, one with two classes and the other with three classes. The three-class solution produced classes (patient groups) and descriptions of relevant features that matched three medically established syndrome categories. The two-class solution consisted of one group of patients having one syndrome and another group containing a mixture of patients having one or the other of two syndromes.

The presented conceptual clustering method has also been used to build classifications of 12 microcomputers [Michalski and Stepp, 1983] and a collection of 100 Spanish folksongs [Stepp, 1980].

5.9. Performance analysis

The search problems inherent in this method of conceptual clustering cannot be solved using exhaustive techniques because of the overwhelming computational complexity. The use of heuristic search permits the algorithm to produce near-optimal solutions obtained with limited amounts of computation. Unfortunately, the use of heuristic search makes analysis of algorithm performance exceedingly difficult since the amount of work performed varies with the patterns in the data as they influence the application of the heuristics. The following short discussion analyzes the performance of the CLUSTER/2 program empirically.

CLUSTER/2 is a program written in PASCAL which performs conjunctive conceptual clustering on unstructured events described by vectors of attribute values, as described in this chapter. For the analysis of performance, the problem of rediscovering classes of soybean disease is used to exercise the program. The study relates the overall run-time of the program to the three following program variables:

1. the problem size (number of events and number of attributes),
2. the number of classes formed,
3. the stopping criterion composed of the parameters *base* and *probe*.

The effect of these program variables on performance has been measured empirically by collecting statistics from the program run on various amounts of the soybean disease data with various settings for control parameters. The trends reported seem to be reasonable expectations of the performance of the algorithm. All run times reported are based on runs obtained on a CDC Cyber 175 running the NOS operating system.

In the problem of building a two-class classification of diseased soybeans, the *PRO search* parameters such as *search-probe* and the maximum number of branches at each interior node in the search tree (see Sec. 5.4) had a minor effect on the performance. The overall effect of these parameters (both individually and collectively) on the run time was sublinear for *search-probe* varying from 2 to 8 and the maximum number of branches at each node varying from 1 to 4. Although this empirical result applies only to the soybean data with k=2, the findings are reasonable since these search parameters increase the number of hypothetical clusterings considered approximately linearly, but some of the complexes and clusterings considered are duplicates that are discarded.

5.9.1. Problem size versus run time

To evaluate the effect of problem size on the performance, the soybean disease problem was run with 24 and 12 selected events from the original 47 events. The selected events were

taken equally from the four classes of diseases which humans identify in the data in an attempt to maintain the same underlying conceptual interpretation of the data, Other control parameters were kept constant across all runs. Fig. 5.9 shows the run times for various numbers of events for the first level clustering problem in which clusterings for k=2, k=3, and k=4 are constructed. The influence of the number of events on the run time is judged to be linear. Since the process time per generated complex increases linearly with the number of attributes, the influence of number of attributes on the total run time is also linear. Thus, the overall run time of the algorithm is roughly proportional to the product of the number of events times the number of attributes used to describe each event.

Number of events:	47	24	12
Run time (sec):	92.8	46.1	27.0

Figure 5.9. Run times for various numbers of events.

5.9.2. Number of classes versus run time

The run time is also affected by the range of number of classes constructed. Here, an analysis is presented of the time required to complete one clustering for a given number of clusters k. In the *direct* method used in CLUSTER/2, all k clusters are created and then "juggled" (by NID, see Sec. 5.2.5) to make them disjoint. The amount of computation needed to do this must rise as the number of clusters rises. Fig. 5.10 shows some computation times for producing clusters with various numbers of classes. Because heuristics are involved, the number of iterations of various parts of the algorithm vary in an unpredictable fashion with the number of classes formed. Judging from Fig. 5.10, the effect of number of classes on run time appears to be of higher than linear order (perhaps quadratic). A quadratic effect would be reasonable since the number of stars built is generally proportional to the number of classes, and the computation time per star is roughly proportional to the number of logical multiplication steps

performed which goes up in approximate proportion to the number of classes.[5]

5.9.3. Stopping criterion versus run time

The stopping criterion has a major influence on the performance of the algorithm, but it is hard to interpret its effects. The stopping criterion is specified in terms of the parameters *base* and *probe*. *Base* determines both the number of initial iterations and the number of alternative results reported. It has little effect on run time. *Probe* determines how many iterations are performed looking for a new locally optimum clustering. If a true optimum clustering is achieved, then the *probe* number of iterations performed beyond it are wasted—they just add to the computational expense. If a higher *probe* value causes the probe iterations to reach a new local optimum, then the computation cost is somewhat offset by the achievement of an improved result. The soybean disease problem was run for various values of the *probe* parameter for the two class solution. In this problem, the best solution is obtained in iteration 3, even when the algorithm is made to perform a total of 11 iterations (as it did with probe set to 8). Thus for this problem, a higher value of *probe* yielded no improvement in result and a linearly rising computational cost. This behavior is not at all predictable. Fig. 5.11 presents the run times obtained for the soybean disease problem with various values for *probe*. When taking this data, the number of classes was two.

Number of classes:	2	3	4
Time for 12 events:	3.3	5.7	15.9
Time for 24 events:	6.1	10.0	30.0
Time for 47 events:	13.1	34.7	45.0

Figure 5.10. Run times in seconds for various numbers of classes and events.

5. The star G(e,F) typically requires a multiplication step for each event in F. With k classes, F contains k-1 events.

Probe value:	1	2	4	8
Run time (secs):	9.5	12.3	18.1	28.9

Figure 5.11. Run times for different probe values.

5.10. The effect of parameter β on optimal class number

The measure of the total sparseness of the solution is used to judge the best number of classes to form at each level of the hierarchy. Data from the clustering of soybean disease cases (with base=1 and probe=1) for k=2 through k=6 are summarized in Fig. 5.12. As k increases, the sparseness usually decreases because events are partitioned into smaller complexes that can fit the data better. On the other hand, increasing k is undesirable as it raises the complexity of the clustering. The measure used that reflects this trade-off is

$$S = \text{sparseness} \times k^{\beta}$$

where β is a parameter that balances the influence of k versus the sparseness. Fig. 5.12 shows values of parameter S for β=0 through β=4 and k=2 to k=6 for the clusterings of soybean diseases. For low values of β, the best clustering indicated by the smallest value of S is with k=5. As the importance of number of classes is increased by increasing the value of β, the best clustering shifts to k=4 at β=3. By extrapolating the values in Fig. 5.12, the best number of classes to form shifts to 2 when β is set to 7. The value of β=3 is the default setting that appears to give good balance to the number of classes versus the fit.

Number of clusters	Sparseness ($\times 10^5$)	Parameter S fit vs. complexity β=0	β=1	β=2	β=3	β=4	Cpu time used (on Cyber 175)
2	28.0	28.0	56.0	112.	224.	448.	9.5 sec
3	5.90	5.90	17.7	53.1	159.	478.	28.7 sec
4	0.39	0.39	1.56	6.24	25.0	99.9	34.6 sec
5	0.22	0.22	1.10	5.50	27.5	138.	35.5 sec
6	0.37	0.37	2.22	13.3	79.9	480.	53.2 sec

(Data values obtained from runs of 47 soybean diseases with base=1 and probe=1.)

Figure 5.12. Values of sparseness, S, and cpu time for different values of k.

CHAPTER 6

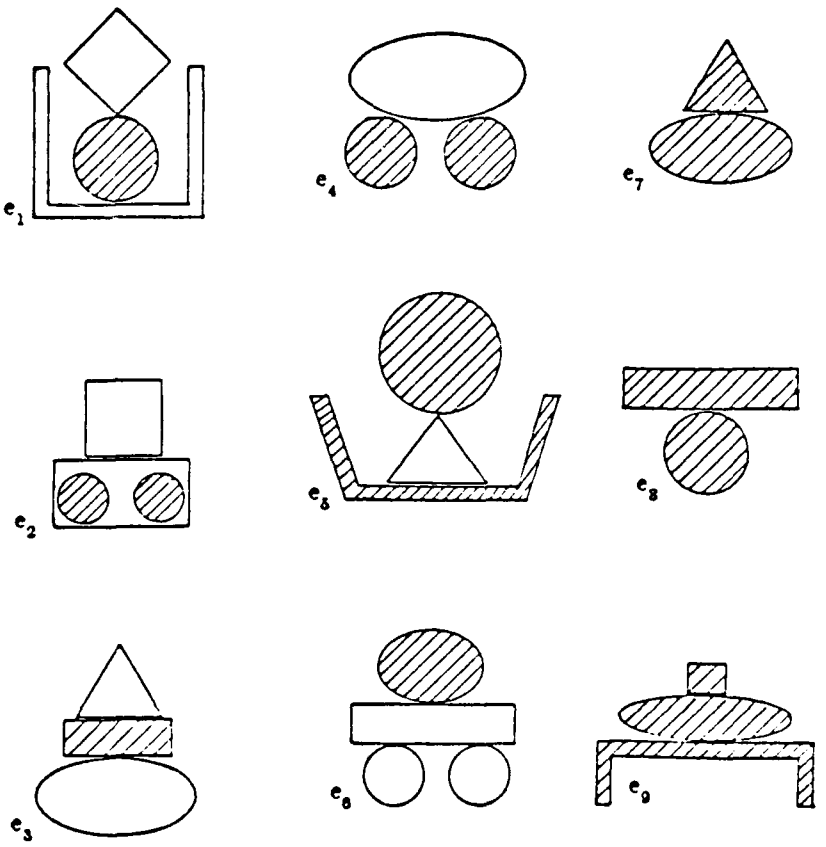
METHOD 2: CLUSTERING STRUCTURED OBJECTS

6.1. Overview

This chapter presents the methodology for building classifications of examples which are described by conjunctive statements in first-order logic. The specific representational scheme is an extended predicate calculus called Variable-valued Logic system 2 (VL_2) [Michalski, 1980a] which is a subset of the more recent language, Annotated Predicate Calculus [Michalski, 1983], described in Chapter 4. The VL_2 representation has much more expressive power than the attribute representation used in the previous chapter.

When considering problems involving structured examples, one usually thinks of Winston's blocks-world, and his pioneering work on learning discriminant descriptions of structures (e.g., arches) [Winston, 1975]. Consider the nine blocks-world examples and selected corresponding VL_2 descriptions shown in Fig. 6.1. Object parts are represented by existentially quantified variables. (If the explicit quantification is omitted, it is always implied.) It will be shown later that it is the existential quantification that makes the conceptual clustering problem difficult when handling structured objects.

The mechanisms for performing inductive inference on VL_2 statements are graph-theoretic. It is therefore important to review the graph representation semantics that underly the VL_2 statements.



e_1: ∃ p1,p2,p3, [ontop(p1,p2)] [ontop(p2,p3)] [size(p1)=medium] [size(p2)=medium] [size(p3)=large] [texture(p1)=clear] [texture(p2)=shaded] [texture(p3)=clear] [shape(p1)=diamond] [shape(p2)=circle] [shape(p3)=ushape] [inside(p2,p3)].

e_4: ∃ p1,p2,p3, [ontop(p1,p2)] [ontop(p1,p3)] [next(p2,p3)] [size(p1)=large] [size(p2)=medium] [size(p3)=medium] [texture(p1)=clear] [texture(p2)=shaded] [texture(p3)=shaded] [shape(p1)=ellipse] [shape(p2)=circle] [shape(p3)=circle].

e_7: ∃ p1,p2, [ontop(p1,p2)] [texture(p1)=shaded] [texture(p2)=shaded] [shape(p1)=triangle] [shape(p2)=ellipse] [size(p1)=medium] [size(p2)=medium].

Figure 6.1. Nine structured objects and selected corresponding descriptions.

6.2. Representing VL_2 statements

6.2.1. Lexical representation

The VL_2 language that is implemented in the program INDUCE/3[1] is defined by the 5-tuple (V,F,S,R,I) where:

V is a set of variable symbols. Each variable symbol x_i is associated with a domain $D(x_i)$. A group of variables which have the same domain are labeled with the same variable symbol with different subscripts, e.g. $y_1,y_2,...,y_k$, $z_1,z_2,...,z_l$ are specifications of variables in two variable groups that assume values from two domains denoted $D(y)=D(y_i)$ and $D(z)=D(z_i)$.

F is a set of n-ary function and predicate symbols. Each n-ary function symbol represents a mapping from an argument space into the domain of the function. For a function $f(x_1,x_2,...,x_n)$, this is a mapping:

$$D(x_1) \times D(x_2) \times ... \times D(x_n) \rightarrow D(f)$$

where $D(x_1)$, $D(x_2)$, ... ,$D(x_n)$, $D(f)$ represent the domains of the variables x_1, x_2, ... x_n and the domain of the function f, respectively. A predicate is a function whose domain is the set of values {FALSE,TRUE} or equivalently {0,1}.

S is a set of symbols including:

$$(\ ) \ [\ ] \ = \ < \ > \ \neq \ \geq \ \leq \ \Rightarrow \ \wedge \ \vee \ \exists \ , \ .$$

R is a set of formation rules as follows. Formulas are used to describe situations, and also as parts of implicative statements. The formulas are defined by the following rules:

1. A selector is a formula.
2. If V, V_1 and V_2 are formulas, then so are:

---

1 INDUCE/3 is an extended version of INDUCE/1 [Larson and Michalski, 1977, Larson, 1977] and INDUCE/2 [Hoff, Michalski, and Stepp, 1983].

(V)	a formula in parentheses
¬V	inverse
$V_1 \wedge V_2$ or V_1V_2	conjunction
$V_1 \vee V_2$	disjunction
$\exists\ x_1,x_2,...,x_k$, (V)	existential quantification

The form of a *selector* is either

[*atomic-function* REL *value-of-atomic-function*]

or

[*predicate-function*].

The form [*atomic-function* REL *value-of-atomic-function*] (e.g., [color(box)=red∨blue]) is defined in Chapter 4. The form [*predicate-function*] (e.g., [ontop(box_1,box_2)]) is equivalent to [*predicate-function*=1] (i.e., [*predicate-function*=TRUE]). Below are some examples of selectors:

Selector	Interpretation: (TRUE if)
$\exists$ $wall_1$, [color($wall_1$) = white]	The color of the wall "$wall_1$" is white.
$\exists$ box_1, [length(box_1) > 1]	The length of the box "box_1" is greater than 1.
$\exists$ box_1, [weight(box_1) = 2..5]	The weight of box "box_1" has a value between 2 and 5, inclusively.
$\exists$ x_1, x_2, [ontop(x_1,x_2)]	The part "x_1" is on top of the part "x_2".

I is a set of interpretation rules that follow the well established logical interpretations given to negation, conjunction, disjunction, and existential quantification [Hoff, Michalski and Stepp, 1983].

6.2.2. Graph representation

A VL_2 formula can be represented as a *colored*[2] graph with labeled nodes and directed labeled edges. The labels on the nodes can be either a selector containing an *n-ary descriptor* without its argument list, or a quantified variable. The edges are optionally labeled with integers 1,2,... that refer to the position of the argument at the head of the edge. The label is omitted if the argument position is irrelevant.

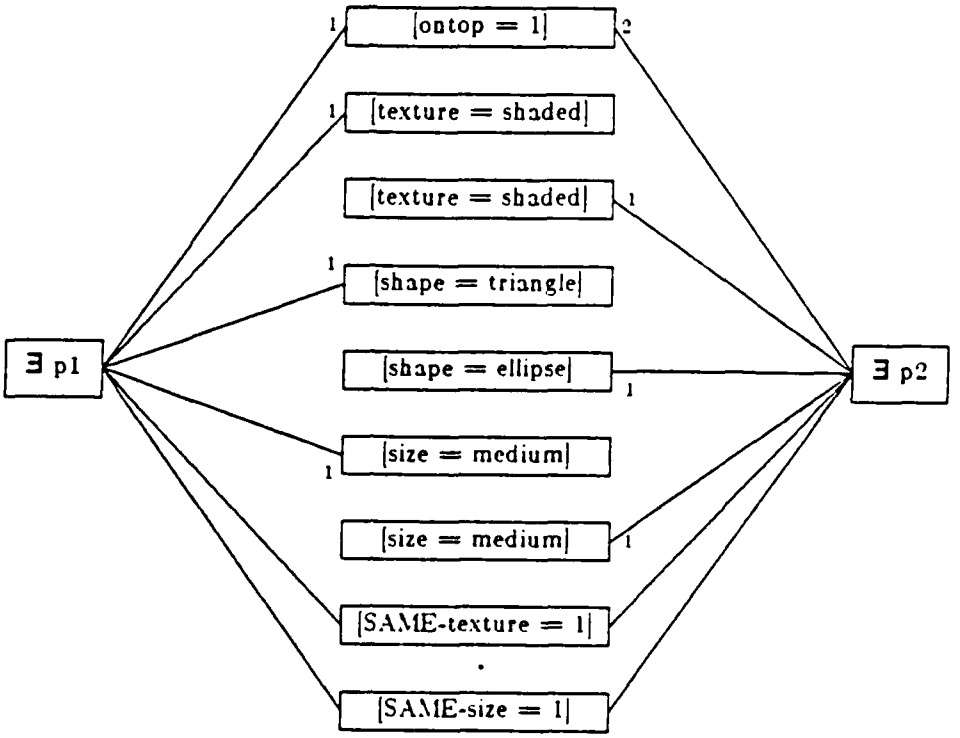
For simplicity, the graph structure for conjunction and disjunction of selectors is omitted from the graphs. Each graph is interpreted as the conjunction of all selector nodes in the graph. A disjunctive description having one or more disjunctions of conjunctions of selectors is represented as the disjunction of two or more graphs. Thus, a graph structure contains just selector nodes and existentially quantified variable nodes. It is adequate for representing one VL_2 complex (a conjunctive description). As an example, the statement describing event e_7 from Fig. 6.1 is shown in Fig. 6.2 as a graph. The logical evaluation of the graph is the conjunction of all selectors shown in the the middle.

If V_1 and V_2 are VL_2 statements, implicative and equivalence statements may be formed as

$$V_1 \Rightarrow V_2 \quad \text{and} \quad V_1 \Leftrightarrow V_2.$$

V_1 is called the *condition* and V_2 is called the *consequence*. Each is represented by a separate graph or a disjunction of graphs. At present, the consequence must be a single conjunctive statement. In an implicative statement, if V_1 is true then V_2 is asserted; if V_1 is not true then nothing is known about V_2. In an equivalence statement, if V_1 is true then V_2 is asserted; if V_1 is not true then the negative of V_2 is asserted. To make the negative assertion unambiguous, V_2 is restricted to be just a single selector in equivalence statements. The negative of such a single selector $[x_i = \textit{value-list}]$, where x_i is a function symbol and *value-list* is the internal disjunction of possible values the function can take, is the selector $[x_i = D(x_i) - \{\textit{value-list}\}]$. That is, the

2 A colored graph has nodes and links that match only if they have corresponding link-color and node-color labels



Graph of e_7: $\exists\ p_1, p_2$, [ontop(p_1,p_2)][texture(p_1)=shaded][texture(p_2)=shaded] [shape(p_1)=triangle][shape(p_2)=ellipse][size(p_1)=medium] [size(p_2)=medium][SAME-texture(p_1,p_2)][SAME-size(p_1,p_2)]

Figure 6.2. The graph representation of example e_7.

selector is negated by asserting that variable x_i takes a value in the complement of the *value-set* with respect to the domain of x_i.

6.3. Comparing two complexes by graph matching

In the process of evaluating generalized class descriptions it is necessary to logically compare two complexes. Given two complexes α_1 and α_2, the possible questions to ask are

- do α_1 and α_2 intersect?

  (i.e., does there exist any example which could satisfy both α_1 and α_2?).

- does α_1 completely cover α_2?

  (i.e., do all examples that satisfy α_2 also satisfy α_1?).

Note that examples are represented in the same language and graph structure as complexes. Thus, with α_2 replaced with a description of an example, the second test can determine if a particular generalized description α_1 covers a particular example e. Tests to answer both of the above questions are minor variations on a *colored* graph matching algorithm, outlined below.

Given are two graphs Q_1 and Q_2, each in the form illustrated in Fig. 6.2. On successful match, every node of Q_1 will be uniquely assigned to a matching node of Q_2. A node of Q_1 matches a node of Q_2 if it has the same *color* and is linked via links of the same *color* to other matching nodes. The *color* of a node is the combination of the type (selector or quantified variable), the function name, and the list of function values. The *color* of a link is the link ordinal that specifies argument order. If no ordinal is present, the link has the *color* 0.

Each conjunctive VL_2 complex may contain selectors involving zero-place functions (i.e., simple attributes) plus one or more groups of *connected* selectors involving single- or multi-place functions. *Connected* selectors are those whose graph is completely connected, i.e., an efficient traversal exists of the entire graph from any starting node. The process of matching one complex with another involves matching the zero-place selectors and then making a traversal of each connected graph while accumulating pairs of matching nodes.

The way a selector is matched depends on the question being asked. If the question is that of intersection, a pair of selectors are matching if they have the same function symbol and at least one common value in their value lists. If the question is that of determining if α_1 completely covers α_2, a selector S_1 in α_1 matches selector S_2 in α_2 only if S_1 and S_2 contain the

same function symbol and all values in the value list of S_2 are also present in the value list of S_1.

After finding matching pairs for all zero-place selectors (i.e., those involving zero-place functions), a count of selectors in Q_2 is made by function symbol. If Q_1 contains more instances of a particular function symbol than does Q_2, the graph match is impossible.

Next, the node with the most colored links is found and becomes the root node of a spanning tree of Q_1. If all links are uncolored (i.e., colored 0), the spanning tree is headed by the node with the greatest number of links. If the spanning tree fails to encompass all nodes (in the case of graphs that are not completely connected) the process is repeated for the remaining nodes. Each secondary spanning tree is attached to the end of the primary tree so that one tree traversal will visit all nodes (other than the zero-place selectors) in the complex.

The graph Q_1 is traversed according to the spanning tree. For each node of Q_1 a matching node of Q_2 is sought. A node in Q_2 matches one in Q_1 if the Q_2 node is not matched with any other node, it has the same function symbol or quantified variable symbol, all links connect to nodes that have been matched, and (for selector nodes) the value list is correct depending on the test for intersection or for complete coverage as described earlier. The nodes of Q_2 are accessed in internal data structure sequence (i.e., not by graph link) in search of the first candidate node. Afterwards, nodes in Q_2 are accessed by links corresponding to matching Q_1 links. When a matching node in Q_2 cannot be found, backtracking along the spanning tree is performed until either failure occurs (all possible subgraphs of Q_2 have been considered) or a previous state involving uncolored links is uncovered and the tree traversal proceeds again with a different match of links.

On successful graph match, a list of matching node pairs is available. This list includes all nodes in Q_1 and a subset of those in Q_2. The internal state of the traversal of the spanning tree over Q_1 is checkpointed so that the search can be resumed if it is desired to search further for additional isomorphisms between Q_1 and Q_2. Thus, this algorithm can produce either a boolean

match or *no match* output, or complete node pairings for one or more matches of Q_1 onto subgraphs of Q_2. By preplanning the graph search starting with a node with the most colored links, the amount of backtracking is minimized.

6.4. Structure template matching

The graph match algorithm of the previous section is often used to map one structure template (given by a covering complex T) onto each isomorphism found in a set of complexes E. The graph match identifies corresponding quantified variables in the template and in the example. It is then possible to recode all examples in an attribute/value form without quantified variables.

This process begins by assigning unique dummy attributes x_1, x_2,... to each zero-place function and each single- or multi-place function in T. Then T is matched with an example e in E. If there is no match, no recoding of e is performed—e is skipped. If there is a match of graphs, then e is recoded in terms of the dummy attributes to form an unstructured complex of the form

$$[x_1 = A_1][x_2 = A_2] \ldots [x_n = A_n]$$

where a value list A_i is the same as the value list in e for the selector which is graph matched onto one in T assigned to dummy variable x_i.

If additional isomorphisms between T and e are found, each one yields an independent attribute-based unstructured complex. When this is completed for all examples in E, a complete attribute-only description of E under the constraint of structure template T has been generated. This process is handy for generating attribute descriptions of examples for use in generalizing only the value list portions of graphs without changing the graph structure. After such a generalization, the generalized value lists can be used to replace the value lists in corresponding selectors in T.

As an example, let T be the structure template

[ontop(p1,p2)][size(p1)=mediumVlarge][texture(p2)=shaded]

and let E be composed of the three examples e_1, e_4, and e_7 from Fig. 6.1. The following four unstructured complexes are produced by structure template matching with attributes x_1 assigned to *ontop*, x_2 assigned to *size*, and x_3 assigned to *texture*.

1: $[x_1 = 1][x_2 = \text{medium}][x_3 = \text{shaded}]$

2: $[x_1 = 1][x_2 = \text{large}][x_3 = \text{shaded}]$

3: $[x_1 = 1][x_2 = \text{large}][x_3 = \text{shaded}]$

4: $[x_1 = 1][x_2 = \text{medium}][x_3 = \text{shaded}]$

Complex 1 above is derived from example e_1. Complexes 2 and 3 are derived from example e_4 where the template matches in two different ways. Complex 4 is derived from example e_7.

6.5. Repeated discrimination

The algorithm developed for unstructured events in Chapter 5 is modified for application to structured events. The fundamental approach is the same: classifications are constructed for k varying from k_{min} through k_{max} by iteratively selecting k seed events and then using inductive inference (learning from examples) to generate descriptions for k classes. As necessary, multiply covered events are reassigned to a class by the procedure NID (suitably modified for structured events). When this approach is followed, the resulting class descriptions may be *weakly intersecting*, i.e., the description spaces intersect but not at the points representing observed events. For some uses, this intersection is undesirable. The approach presented in Chapter 7 uses a method that always generates disjoint class descriptions.

The algorithm using repeated discrimination to generate classifications for structured examples is presented here in a fashion that parallels the description of the approach for unstructured examples in the previous chapter. The following subsections describe the similarities and differences between the methods as they relate to criterion evaluation, generating

conceptual covers, applicable generalization transformations, building a star, and making intersecting classes disjoint (or at least only weakly intersecting).

6.5.1. Criterion evaluation

As with the method used for building classifications of unstructured events in Chapter 5, the classification goal is specified by combining elementary evaluation criteria in a user-specified LEF. As described in Chapter 5, the LEF gives an ordered list of elementary criterion functions which are to be used and the tolerance associated with each function. The elementary criteria implemented in the program INDUCE/3 are:

(1) The number of positive examples that are covered by a class description.

(2) The number of selectors in a class description.

(3) The number of negative examples that are covered by a class description.

(4) The total cost of all functions contained in a class description

(5) The total cost of all quantified variables used in a class description.

6.5.2. Generating conceptual covers

For unstructured examples, covers of selected examples are generated by the REFUNION operation (Sec. 5.2.2). Given a set of examples represented as complexes, the REFUNION of the complexes is the most specific generalization that covers all examples. It is easy to compute REFUNION for unstructured examples when all examples are described by identical attribute lists. Finding the REFUNION for a set of structured objects is much more difficult, since an object may contain unique subparts that have no corresponding part in any other object, or it may have several subparts that are similar in multiple ways to some part of another object.

In terms of the graph-theoretic representation of VL_2 complexes, the REFUNION of a set

of such complexes is the maximal[3] graph which is isomorphic to some subgraph of each complex. Finding such a graph is computationally expensive. A method for generating a quasi-optimal (quasi-maximal) REFUNION of a set of complexes C is summarized in the diagram of Fig. 6.3. The process begins by developing a number of quasi-maximal graphs by adding selectors one at

Given: C, the collection of complexes to be covered,
the evaluation criterion LEF

- Pick a seed event c ∈ C
- Generate set R^c composed of single-selector complexes as the first partial REFUNION
- Empty set R
- Repeat until R contains NCONSIST complexes
  - Generalize all value lists to '*'
  - Trim complexes in R^c to VLMAXSTAR
  - Make ALTER many new offspring complexes out of each parent complex in R^c by adding one new selector
  - Discard complexes which become too specific; save in set R any parent complex whose offspring are all too specific
- Make a maximally specific generalization of the value lists in complexes in R
- Use LEF to select the best REFUNION complex

Figure 6.3. An algorithm for generating a REFUNION of structured examples.

3 A maximal graph is one with the greatest number of nodes.

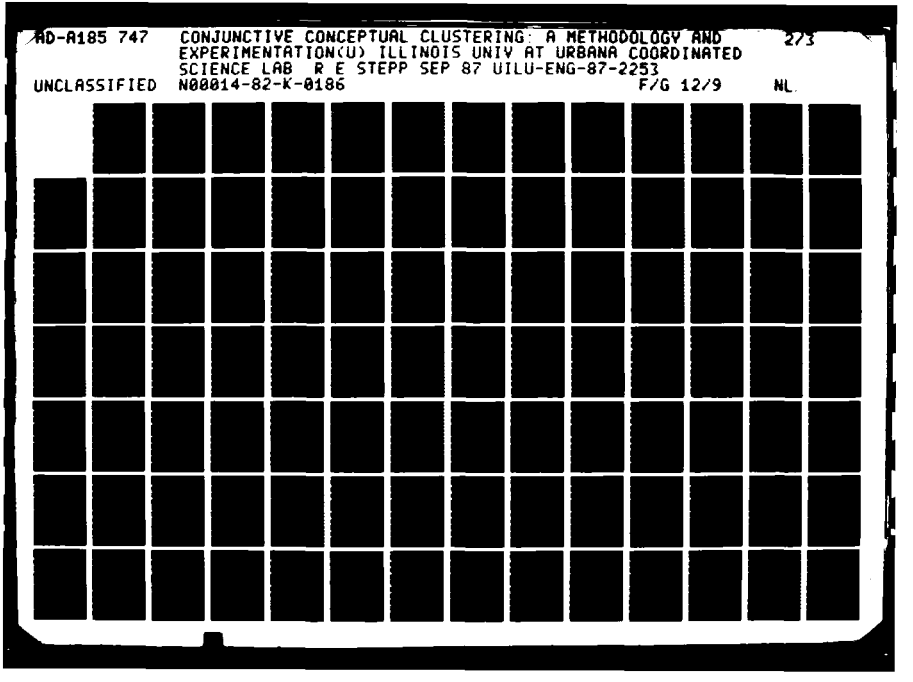
a time until subsequent additions would overly specialize the graph such that it would not cover all complexes in C. Then the reference lists of the selectors in the graph and those in zero-place selectors are generalized by applying the various value set transformations of Chapter 4. The generalizations are performed to a degree that balances the desire for simple value lists with that for forming a maximally specific description (e.g., a list of values in a linear domain would be generalized to the single minimal enclosing interval).

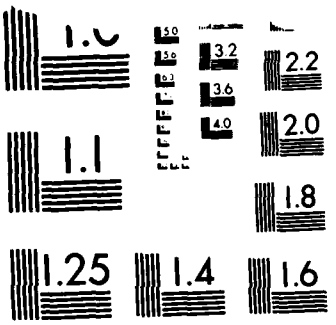
Given to the REFUNION procedure are a set of complexes to be covered C and an evaluation criterion LEF. One complex e in C is arbitrarily selected as the *seed*. The first set of candidate REFUNION complexes are generated to form the set R^c. Each complex consists of a single selector extracted from the graph for e. Next, an iterative process is performed to *grow* the candidate complexes, one selector at a time, until any further growth makes them too specific.

Each iteration involves four steps. First, all candidate complexes in R^c are made more general by replacing all value lists in selectors with the "don't care" symbol * which denotes the whole domain of the function. Second, the complexes are trimmed[4] and duplicates eliminated such that only the VLMAXSTAR (user-defined parameter) number of complexes remain in R^c. Then third, each surviving *parent* complex is used to generate ALTER (user-defined parameter) number of *offspring* complexes. Each offspring contains one more selector than the parent. The offspring are given different selectors from e that are not not in the parent complex. Selectors to add from e are chosen according to their *connectedness*[5] to the parent. The offspring complexes replace the parent in set R^c. The fourth and final step of each iteration [illegible] created offspring complexes, eliminating all that fail to cover all complexes [illegible] offspring of one parent are so eliminated, the parent complex [illegible]

4 Trimming is done to limit the combinatorial explosion that [illegible] were considered.

5 The connectedness of two selectors is the number [illegible] function part of the selectors.


AD-A185 747 CONJUNCTIVE CONCEPTUAL CLUSTERING: A METHODOLOGY AND EXPERIMENTATION(U) ILLINOIS UNIV AT URBANA COORDINATED SCIENCE LAB R E STEPP SEP 87 UILU-ENG-87-2253
2/3
UNCLASSIFIED N00014-82-K-0186 F/G 12/9 NL



result set R. When all offspring are too specific, the parent is maximally specific.

When the above iterations are completed, set R contains a number of quasi-maximally specific complexes, all of which cover all complexes in C. The size of set R is given by the parameter NCONSIST. Note that the value lists in the selectors are still the "don't care" value * at this point.

The value lists are replaced by new values that are generalized only to the extent necessary to cover all complexes in C. The following processing is applied to every complex r in set R. The structure template match algorithm of Sec. 6.3. is used to generate attribute-based descriptions of all complexes in C. Set B will denote the resulting attribute-based complexes corresponding to complexes in set C. Starting with a working complex in which the value lists of all selectors are empty, each complex b in B that was derived from the same concept in C is merged with the working complex by forming the union of values in value sets of selectors in the working complex and those in b. Value set transformations are applied to the merged complex using the GENRLIZ procedure as described in Sec. 5.2.3. The change in amount of generalization caused by each merge operation is measured and the complex which gives rise to the minimum amount of generalization is selected. The other complexes considered are discarded. Each complex in C which is covered by the working complex is marked and each complex in B which was derived from a marked complex in C is also removed. The process repeats with the working complex set at the state following the value list merge for the selected complex from B. When all complexes in B are eliminated, the working complex contains value lists that are generalized the minimum amount to cover all examples in C. These value lists are returned to graph r that was previously used as the template complex. The evaluation LEF is used to pick the best complex from set R after each has undergone the value list processing just described.

6.5.3. Generalization transformations

All generalization transformations described in Sec. 4.2 are implemented in the program INDUCE/3 whose inference procedures are used to carry out the conceptual clustering algorithms involving structured objects. The descriptor construction transformations are provided as a prelude to inductive inference so that the constructed descriptors augment the complexes describing examples. The process is divided into three passes over the complexes.

The first pass applies the transformations for *generating chain properties rule* and *generating equivalence relations rule*. For example, when this pass is completed for the first example e_1 in Fig. 6.1, the following selectors are added to the graph representation of the complex describing e_1:

[MOST-ontop(p1)][LEAST-ontop(p3)][MOST-inside(p2)][LEAST-inside(p3)] ∧

[SAME-size(p1.p2)][SAME-texture(p1.p3)]

The second pass applies background knowledge inference rules to the examples. The graph matching procedure is used to compare the left hand side of each background knowledge rule to the description of each example. When a match is found, the selectors in the right hand side of the background knowledge rule are appended to the example description (unless they are already present) using the same binding of quantified variables establish by the graph match. The background knowledge rules are recursively applied until no further matches to examples are found.

The third pass over the example descriptions is to add zero-place descriptors for *counting quantified variables*, *counting similar selectors*, *counting distinct references*, and adding *universal properties*. For example, when this is done for example e_7 from Fig. 6.1, the constructed zero-place selectors are

[#parts=2][#parts(texture=shaded)=1][forall-parts(texture=shaded)][#diff-textures=1] ∧

[#parts(shape=triangle)=1][#parts(shape=ellipse)=1][#diff-shapes=2] ∧

[#parts(size=medium)=2][forall-parts(size=medium)][#diff-sizes=2]

Note that the "forall" predicates, which may be added by application of the *universal properties* descriptor construction transformation, provide a way to capture limited universal quantification semantics while still keeping the simple graph representation that has only selector and existentially quantified variable nodes.

The several dimension-reducing and relevant value set transformations are applied during star generation. They are described below.

6.5.4. Building a star

The process of generating a star from structured events is somewhat different than the process described in Chapter 5. A star, denoted by G(e|F), is a set of consistent class descriptions each of which covers event e and none of the events in event set F. The procedure begins by forming a partial star $G^P(e)$ which is a set of class descriptions each consisting of one unique selector from the description of event e. Each description in G^P covers event e and possibly one or more events in set F, i.e., some descriptions in G^P may not be consistent with respect to F.

At this point each description in G^P consists of just a single selector. The process described in the next paragraph is performed iteratively to specialize the descriptions by conjoining one additional selector on each iteration. Any generated descriptions which are consistent (i.e., do not cover any events in F) are placed on a list of alternative descriptions. The process continues until a preset number of consistent alternatives have been generated, or until all possible selectors from event e have been conjoined. The final value of G(e|F) is the set of complexes in the resulting list of alternatives.

On each iteration each description α in $G^P(e|F)$ is compared with the description of event e. A list of predicates and functions that are in e but not in α is generated. A limited number

of these predicates and/or functions is selected and their corresponding selectors are conjoined separately to complex α to produce different specializations of description α. The number of predicates and/or functions selected (and hence the number of alternative specializations) is given by the parameter ALTER. The order of selection of predicates and/or functions is by decreasing *connectedness* to predicates and/or functions already in α. The connectedness of two predicates or functions is the number of arguments they have in common.

After generating new descriptions, the partial star contains ALTER times more candidate descriptions than before. To limit the combinatorial explosion, each new partial star is trimmed down to just a few of the best descriptions, as determined by the evaluation criterion given by the LEF. The number of descriptions retained in each partial star is given by the parameter VLMAXSTAR.

When a set of consistent descriptions has been built, they have the property that they cover event e (possibly other events in set E as well) and none of the events in F. Furthermore, the last operation to a class description (i.e., conjoining an additional selector) transforms the description from one that was too general (and inconsistent) to one that is possibly too specific (although consistent). The need thus arises to perform some limited generalization to make the description as general as possible, but still not cover any event if F. The generalization considered is that of generalizing the value list portion of one or more selectors. This is done using the (unstructured) techniques of Chapter 5, specifically the star generation algorithm for unstructured examples. It is in this stage too that the value list transformations of *dropping condition*, *adding alternatives*, *closing the interval*, and *climbing the value hierarchy* are preformed where appropriate. Given a complex α to generalize, value list generalization proceeds in the following way.

(1) A modified complex α' is created from complex α by replacing all value lists in selectors with the *don't care* symbol "*" (this symbol denotes all possible values in the domain of the variable).

(2) All subgraphs of all examples that match the graph of α' are found. Each isomorphism yields a derived positive event. This is done by assigning a dummy attribute to each selector in complex α' and then generating attribute/value pairs for each selector in the example subgraph, as described in Sec. 6.3. For each attribute/value pair, the attribute is the one assigned to the corresponding selector of α' and the value is the value list from the selector in the example subgraph.

(3) Using the star generation technique of Chapter 5, the most general (unstructured) description is found that covers an event derived from event e and optionally other derived positive events, but no events derived from negative events in F.

Corresponding selectors that fit a structural template are used to derive a set of positive and negative events that are unstructured. A star is built using derived events, $G(e'|F')$ where e' is an unstructured event derived from e, and F' is the set of all events derived from events in F. This star contains alternative ways to maximally generalize the reference values without covering any event in F'. These reference values are returned to the corresponding selectors in the original (structured VL_2) complex. Since no event in F' is covered, if follows that no event in F is covered. Since at least one isomorphism e' of e is covered, e is covered too. After this step, the value lists are as general as possible, and thus G(e|F) has the *most general* property which is sought.

6.5.5. NID: Making nondisjoint complexes disjoint or weakly intersecting

The star building algorithm generates complexes containing functions and predicates of existentially quantified variables. Each complex is a conjunction of selectors. Such a complex contains neither negation nor disjunction, although a selector may contain internal disjunction and internal negation, as described in Chapter 4. One result of this constraint is that there is no construct to indicate the negative quantification "*there does not exist.*" Complexes constrained to use existential quantification without negation will logically intersect unless made disjoint by

non-intersecting value lists of zero-place descriptors. This is illustrated by considering the two complexes

(1) ∃ part, [size(part)=large]

and

(2) ∃ part, [size(part)=small]

While these statements appear to split the universe into large and small things, they actually intersect. This intersection includes an object containing two parts such as

∃ $part_1$, $part_2$, [size($part_1$)=small][size($part_2$)=large)].

One way to make the class descriptions (1) and (2) become disjoint is to introduce a zero-place selector, such as [#parts=1]. The two descriptions

(3) ∃ part, [#parts=1][size(part)=small]

(4) ∃ part, [#parts=1][size(part)=large]

are now disjoint—no example exists which satisfies both descriptions. This illustrates the importance of zero-place (quantifier-free) descriptors. Such descriptors are created by application of the descriptor construction transformations described in Chapter 4.

It may happen that no disjoint zero-place descriptors are used to describe the classes at hand. In such a case it is not possible to generate disjoint descriptions thought it may be possible to make them become only *weakly intersecting*. Two descriptions are *weakly intersecting* if they intersect only at unobserved events, i.e., if no given example satisfies both descriptions. In the remainder of this section, a method will be described for transforming intersecting descriptions into ones that are disjoint or intersect only weakly.

Given are k complexes α_1, α_2,..., α_k, a set of events E, and seed events e_1,...,e_k for each complex. If no event in E is covered by more than one complex, then the set of complexes is disjoint or only weakly intersecting and the algorithm exits without having to do any work.

All observed events which are multiply covered are put into set M. Then k event sets A_i, $1 \leq i \leq k$, are built with A_i containing all events covered by α_i but not in M. Each complex α_i is replaced by the structured REFUNION complex obtained by covering the example set A_i. Because tne REFUNION complexes are maximally specific, they do not intersect on observed events and are either disjoint or weakly intersecting.

At this point, the events in M are not covered by any complex. Each event in M is assigned to one of the k complexes by completing the following steps.

(1) K augmented complexes α'_i, $1 \leq i \leq k$, are constructed by building the structured REFUNION complex covering A_i and event $m \in M$ being fitted to a complex.

(2) Event m is assigned to the complex α'_j, $1 \leq j \leq k$, which covers m and is best according the the LEF. A_j is replaced by $A_j \cup \{m\}$ and α'_j replaces α_j.

In the end, the α' complexes are output along with a report of any events which could not be covered.

6.6. An example problem: Classifying blocks-world figures

The presented methodology will be illustrated by an example using the blocks-world figures in Fig. 6.1. Annotated input and output statements for the program INDUCE/3 that hosts a prototype implementation of the method are given in Appendix B.

The input for the problem consists of nine VL_2 example descriptions (of the kind shown in Fig. 6.2) and some background knowledge consisting of the types of the variables plus statements defining a value hierarchy for the function *shape* as illustrated in Fig. 6.4.

The initially given function and predicate symbols used to describe the examples and their value domains are:

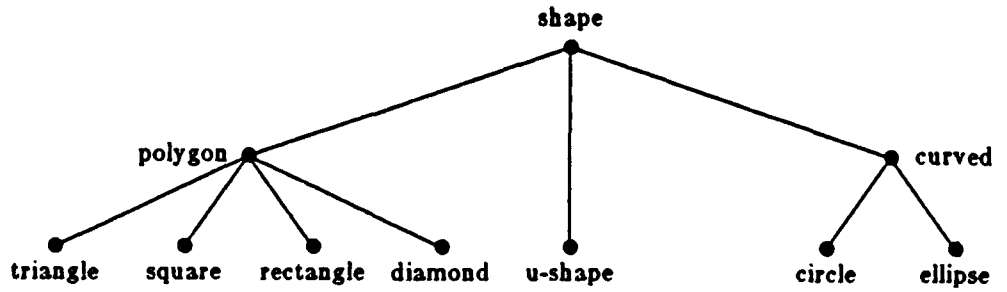


Figure 6.4. The value hierarchy for the domain of the shape function.

Symbol	*Domain*	*Description*
ontop(x,y)	{0,1}	x is ontop of y
inside(x,y)	{0,1}	x is inside of y
next(x,y)	{0,1}	x and y are next to each other
texture(x)	{clear,shaded}	the texture of x
size(x)	{small,medium,large}	the size of x
shape(x)	{triangle,square,rectangle, diamond,circle,ellipse,u-shape, polygon,curved}	the shape of x

The following derived descriptors are added by the application of descriptor construction transformations.

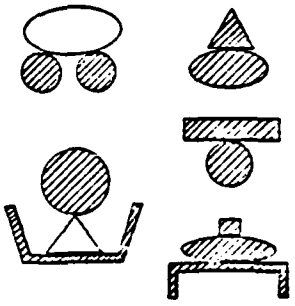
Symbol	*Domain*	*Description*
MOST-ontop(x)	{0,1}	x is a top part
LEAST-ontop(x)	{0,1}	x is a bottom part
MOST-inside(x)	{0,1}	x is an inner part
LEAST-inside(x)	{0,1}	x in an outer part

SAME-texture(x,y,...)	{0,1}	the parts have the same texture
SAME-size(x,y,...)	{0,1}	the parts have the same size
SAME-shape(x,y,...)	{0,1}	the parts have the same shape
#parts	{0..4}	the number of parts
#parts(property)	{0..4}	the number of parts with given property
#diff-textures	{0..2}	the number of different textures
#diff-sizes	{0..3}	the number of different sizes
#diff-shapes	{0..4}	the number of different shapes
forall-parts(property)	{0,1}	property holds for all parts

In this example, no background inference rules are used.

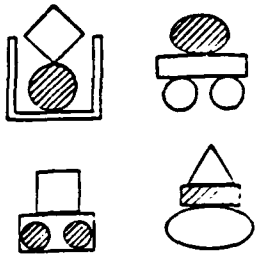
The classifications obtained for k=2, k=3, and k=4 are shown in Figs. 6.5, 6.6, and 6.7, respectively. For this problem involving only nine examples, the classification is not hierarchical. The goal of the classification was to maximize the specificity of descriptions as indicated by the number of selectors they contain. For the sake of clarity, the descriptions shown in Figs. 6.5 through 6.7 are edited to contain only those selector parts which take different values in at least two of the complexes. The conceptual clustering algorithm generates descriptive statements rather than discriminant rules. For the clusters in Fig. 6.5, the full descriptive statement includes about 40 selectors. Except for the selector "#parts(texture=clear)", the values sets of corresponding selectors in the descriptive statements intersect. The complete descriptive statements for all clusters are given in Appendix B.

In Fig. 6.5, the classifications are based on the number of parts of a certain texture. In the first classification, one class contains figures having less than 2 clear parts and the other class contains figures with 2 to 3 clear parts. In the second classification, one class contains figures having exactly one shaded part and the other class contains figures having two or three shaded parts. In each case, the ranges of numbers of parts were determined by developing a conjunctive concept describing each class which was optimized with respect to the goal of the



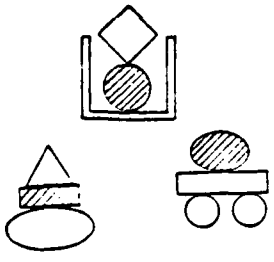
Class 1: [#parts(texture=clear) < 2]

"there are fewer than 2 clear parts"



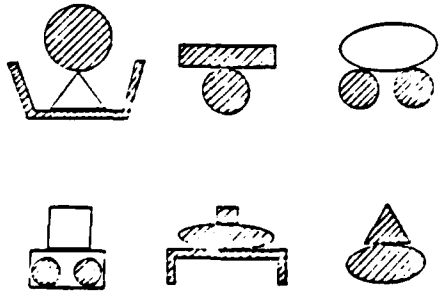
Class 2: [#parts(texture=clear) = 2..3]

"there are 2 to 3 clear parts"



Class 1: [#parts(texture=shaded) = 1]

"there is one shaded part"

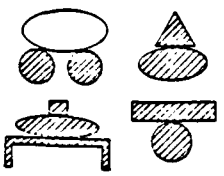


Class 2: [#parts(texture=shaded) = 2..3]

"there are 2 to 3 shaded parts"

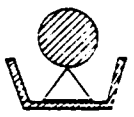
(The LEF is to maximize the no. of events in a class and the no. of selectors in a class description. The two classifications are from the first two iterations of the algorithm.)

Figure 6.5. Two alternative classifications of blocks-world figures for k=2.



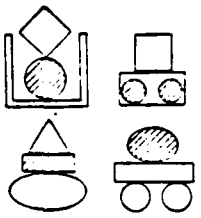
Class 1: ∃ $part_1$, $part_2$, [#parts=2..3] [#parts(size=medium)≠3] [#parts(size=large)≠3] [#parts(texture=clear)<2] [MOST-*ontop*($part_1$)] [*ontop*($part_1$,$part_2$)] [shape($part_1$)≠*circle*] [shape($part_2$)=curved] [texture($part_2$)=shaded]

"there is a top part which is not a circle ontop of a curved part and there are 2 or 3 parts of which fewer than 2 are clear and the medium size parts and large parts number other than 3"



Class 2: ∃ $part_1$, $part_2$, [#parts=3] [#parts(size=medium)=1] [#parts(size=large)=2] [#parts(texture=clear)=1] [MOST-ontop($part_1$)] [ontop($part_1$,$part_2$)] [shape($part_1$)=circle] [shape($part_2$)=triangle] [texture($part_2$)=clear]

"there is a top part which is a circle ontop of a triangle and there are 3 parts of which 1 is medium size, 2 are large, and 1 is clear"

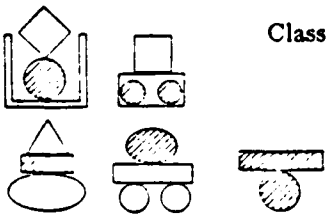


Class 3: ∃ $part_1$, $part_2$, [#parts=3..4] [#parts(size=medium)=2] [#parts(size=large)<2] [#parts(texture=clear)=2..3] [MOST-ontop($part_1$)] [ontop($part_1$,$part_2$)] [shape($part_1$)≠rectangle] [shape($part_2$)≠square]

"there is a top part which is a non-rectangle ontop of a non-square part and there are 3 to 4 parts of which 2 are medium size, less than 2 are large, and 2 to 3 are clear"

Class 1: [#different-shapes=2] [#parts(shape=ushape)=0] [#parts(size=medium)=2] [#parts(size=large)<2]

"2 different shapes are present and there are no ushape parts, 2 medium size parts, and fewer than 2 large parts"



Class 2: [#different-shapes=2..3] [#parts(shape=ushape)<2] [#parts(size=medium)=1..2] [#parts(size=large)<2]

"2 to 3 different shapes are present and there are fewer than 2 ushape parts, 1 to 2 medium size parts, and fewer than 2 large parts"

Class 3: [#different-shapes=3] [#parts(shape=ushape)=1] [#parts(size=medium)<2] [#parts(size=large)=2]

"3 different shapes are present and there is one ushape part, fewer than 2 medium size parts, and 2 large parts"

(The LEF is to maximize the no. of events in a class and the no. of selectors in a class description. The two classifications are from the first two iterations of the algorithm.)

Figure 6.6. Two alternative classifications of blocks-world figures for k=3.

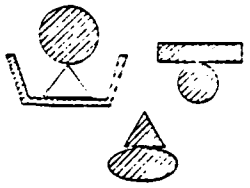
classification as given by the LEF. The descriptions in Fig. 6.5 are disjoint because they are based on non-intersecting value lists of zero-place functions.

In Fig. 6.6, the selectors from the complete class descriptions which take different values in each class are shown along with the selectors for MOST-ontop, ontop, and #parts that serve to bind the variable $part_2$ to a particular part of the blocks-world figure. The first classification is based on the descriptors shape($part_1$), shape($part_2$), texture($part_2$), #parts(size=medium), #parts(size=large), and #parts(texture=clear). The conjunction of selectors using these descriptors is sufficient to form three disjoint classes. Classes 1 and 2 are mutually disjoint because of the #parts(size=medium) and #parts(size=large) selectors. Classes 1 and 3 are mutually disjoint because of the #parts(texture=clear) selector. Classes 2 and 3 are mutually disjoint in a way the program cannot detect. The class descriptions differ on the values of selectors involving shape($part_1$), shape($part_2$), and texture($part_2$). Given the constraints of MOST-ontop, ontop, and #parts, it is logically impossible to build any blocks-world figure that satisfies both class descriptions. The proof of this conjecture is the only means to determine that the class descriptions are disjoint. The program currently is not capable of doing this. It can only compare zero-place function values to determine if descriptions are disjoint in that manner. The second classification in Fig. 6.6 is based on the descriptors #different-shapes, #parts(shape=ushape), #parts(size=medium), and #parts(size=large). Complete descriptive statements for all classes shown in Fig. 6.6 are given in Appendix B.

Fig. 6.7 shows a classification with k=4. There are 17 descriptors that take different values in two or more classes. For ease of presentation, three descriptors that can collectively discriminate between all four clusters have been selected and used in the descriptions shown in Fig. 6.7. The full characteristic descriptions of the clusters are given in Appendix B. The generated class descriptions are mutually disjoint, based on the value lists for selectors containing zero-place functions.

Class 1: [forall-parts(shape=curved)=FALSE] [#parts(size=medium)=0] [#parts(texture=clear)=0]

"objects have no medium size or clear parts and not all parts are curved shape"

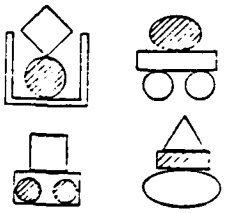


Class 2: [forall-parts(shape=curved)=FALSE] [#parts(size=medium)=1..2] [#parts(texture=clear)=0..1]

"objects have 0 or 1 clear parts, 1 to 2 medium size parts, and not all parts are curved shape"

Class 3: [forall-parts(shape=curved)] [#parts(size=medium)=2] [#parts(texture=clear)=1]

"objects have 1 clear part and 2 medium size parts and all parts are curved shape"



Class 4: [forall-parts(shape=curved)=FALSE] [#parts(size=medium)=2] [#parts(texture=clear)=2..3]

"objects have 2 to 3 clear parts, 2 medium size parts, and not all parts are curved shape"

(The LEF is to maximize the no. of events in a class and the no. of selectors in a class description. The classification from the first iteration of the algorithm.)

Figure 6.7. A classification of blocks-world figures for k=4.

6.7. Performance analysis

The method presented in this chapter for clustering structured events requires more computation power than the method presented in Chapter 5 for clustering unstructured events. As an example of this difference, consider the primitive function for determining if a complex α covers an event e. When dealing with unstructured complexes and events, the attribute vector used in all complexes and events is the same.[6] This makes it easy to compare selector value lists

6. When a complex is printed, selectors with value lists containing all values in their respective domains are not printed. In the implementation, the internal representation of a complex has a slot for a selector involving each attri-

to see if all values for an event are also present in the complex. If so, the complex covers the event. When dealing with structured complexes and events, each event and each complex may be composed of a different combination of descriptors. To compare a complex and an event it is necessary to match the graph of the complex to the graph of the event. This requires much more computation than merely comparing attribute vectors. The net result of the added computational cost of the method for handling structured events is that more emphasis is placed on heuristics. This makes performance trend analysis more difficult.

The implementation in the PASCAL program INDUCE/3 has two layers. The first layer deals with the graph-encoded "rules" (complexes and events) while the second layer deals with the attribute-encoded selector value list parts of rules. For historical reasons the first layer has been called "VL" and the second layer has been called "AQ". Heuristic-based star generation takes place in both layers. In the VL layer, the parameter VLMAXSTAR specifies the number of complexes to retain from one step of star generation to the next. The parameter ALTER specifies the number of different selectors to consider for appending (one at a time) to each of the VLMAXSTAR complexes to specialize it. When NCONSIST number of complexes have been specialized such that they cover no events in the "against" class F, the selector value lists in each complex are generalized by the AQ layer. By the process of template matching (see Sec. 6.4), the AQ layer receives an attribute-list image of the value list parts from selectors of events corresponding to selectors in the complex being considered in the VL layer. The AQ layer performs attribute-based star generation using the heuristic parameter AQMAXSTAR to limit the number of (attribute) complexes that are retained during its star generation process. Although there are four controlling parameters, the parameters controlling breadth of search (VLMAXSTAR, AQMAXSTAR, and ALTER) will be given the same value and their influences evaluated together. The influence of NCONSIST on the performance was evaluated separately and found to be relatively unimportant in its effect on run time. For the blocks-world figures

bute. The value list is encoded as a sequence of binary bits with one bit corresponding to each value in the domain. If all bits are 1's (i.e., if all values in the domain are in the value list) the selector is not printed.

problem, the values of 2, 4, and 8 for NCONSIST made a change in run time of only 1% and made no change in the generated clusterings. This may not be true for other clustering problems.

The following discussion considers the effects on run time of the algorithm by

1. the problem size (number of events and number of descriptors used),
2. the number of classes formed, and
3. the breadth of search during star generation.

6.7.1. Problem size versus run time

To evaluate the empirical relationship between problem size and run time, the blocks-world clustering problem was solved with all 9 events, with 3 eliminated, and with 6 new events added. The run times are shown in Fig. 6.8 for two-class solutions made with ALTER, VLMAXSTAR, AQMAXSTAR, and NCONSIST set to 4. From the times reported in Fig. 6.8, there is an approximately linear effect between 9 and 18 events with the time for 6 events being surprisingly higher than the time for 9 events. A partial explanation is that the heuristics caused the work performed by NID (which accounts for much of the run time) to consist of 8, 6, and 10 star generation steps, respectively for the different problem sizes. When the cost per star generation is estimated by dividing the run time by the number of such steps performed, the estimated star generation times are 5.57, 5.91, and 8.93 seconds, respectively. Thus, the cost of star generation (the most significant computational expense) rises approximately linearly with number of events.

Number of events:	6	9	18
Run time (secs):	44.5	35.1	89.3
Number of stars built by NID:	8	6	10
Time per star (secs):	5.57	5.85	8.93

Figure 6.8. Run times for different numbers of events.

6.7.2. Number of classes versus run time

Noting the evidence presented in Chapter 5 for a quadratic effect by the number of classes on the run time for the problem of clustering unstructured soybean disease events, a similar behavior for clustering structured events would be expected. The data in Fig. 6.9 presents the run times for generating two-class, three-class, and four-class classifications of the nine blocks-world figures. For these runs, the parameters ALTER, VLMAXSTAR, AQMAXSTAR, and NCONSIST all had the value 4. The data show that the effect is more than a linear one, with the run times for k=2 and k=3 suggesting a quadratic behavior. If the times were to typify a quadratic behavior, the time for k=4 should be 135.5 rather than 84.5. It is possible that some diminution in time comes about because there are only 9 events to be divided into 4 classes. In very small classes, some of the techniques (e.g., seed event selection) can exhaust all possible choices and lead to quick solutions. The expected number of stars built by NID should be proportional to k, though this too is greatly affected by the events. Since the time per star generation also rises with k, the net effect is expected to be of second order.[7]

Number of classes:	2	3	4
Run time (secs):	35.1	76.3	84.5
Number of stars built by NID:	6	12	12
Time per star (secs):	5.8	6.4	7.0

Figure 6.9. Run times for various numbers of classes.

6.7.3. Breadth of search versus run time

The three parameters ALTER, VLMAXSTAR, and AQMAXSTAR control the breadth of search conducted through the solution space of complexes generated in the process of building a star. For empirical study, the parameters were given the same values, either 2, 4, or 8. Fig.

[7] The star G(e,F) is built with the 'against set' F composed of k-1 events. The time required to build a star is roughly proportional to the number of events in set F

6.10 shows the run times and number of stars built by NID for the blocks-world problem with k=2. The time per star generation shows an approximately exponential effect in parameter value. The total run time is not necessarily exponential with respect to parameter value. As the search breadth is increased, there is a tendency for the quality of the complexes generated to improve and lead to fewer multiply covered events for NID to handle. Since NID generates a new star for each multiply covered event for each class, reducing the number of multiply covered events by just one leads to a reduction by k in the number of stars generated.

Value for all parameters:	2	4	8
Run time (secs):	25.5	35.1	120.8
Number of stars built by NID:	12	6	8
Time per star (secs):	2.1	5.8	15.1

Figure 6.10. Run statistics for 3 values of parameters ALTER, VLMAXSTAR, and AQMAXSTAR.

CHAPTER 7

METHOD 3: STRUCTURAL TEMPLATE CHARACTERIZATION

7.1. Overview

This chapter presents a methodology for combining parts of the unstructured-example classification building algorithm of Chapter 5 with parts of the structured-example classification building algorithm of Chapter 6. The motivation for this third implementation is the desire to build classifications of structured objects in which class descriptions will be disjoint.

In the method presented in the preceding chapter, the algorithm NID for turning non-disjoint class descriptions into disjoint ones was not actually able to live up to its name. In cases where it is not able to adjust the class descriptions so that each contains a selector for a common zero-place function with non-intersecting value lists, the algorithm can only guarantee to adjust the descriptions to make them weakly intersecting (i.e., intersecting but not at observed events). This is explained and illustrated in Sec. 6.5.5..

The methodology of this chapter is based on a decomposition of the classification building problem into parts: finding an optimized structural characterization of the problem by which quantifier-free attributes are generated, and building a classification based on the generated attributes using the methodology of unstructured classification building of Chapter 5. Because the unstructured descriptions are quantifier free, a disjoint classification will be obtained in all cases[1]. It should be noted that disjoint class descriptions are not required by all classification building problems. When intersecting classes are appropriate, the algorithms of Chapters 5 and 6 may be performed without performing procedure NID. When weakly intersecting classes are

1 Note however that the methodology for generating classifications of unstructured examples may throw out some examples as "anomalies," as described in Sec. 5.2.5

appropriate, the algorithms of this chapter are not required, but they may be more efficient than those of Chapter 6 because much less work is necessary to build classifications of attribute-based examples than graph-based examples. The decomposition by structure template characterization does add additional built-in biases to the solution which may in certain cases make the added costs of the direct approach worthwhile.

The remainder of this chapter will describe the method used for generating the structural template; how it is used to derive attribute-based descriptions of the examples, and how they are presented to the previously described method for building the classification. The chapter concludes with an example problem that has been solved by both the presented algorithm and by humans. In the human solutions, the subjects generated disjoint classes. This is taken as partial evidence that the ability to generate classifications of mutually disjoint classes is necessary in order to attempt to attain some learning skills practiced by people.

7.2. Generating attribute-based descriptions for structured examples

The process of building classifications by structural template characterization is implemented in the program CLUSTER/S. That implementation is the amalgamation of the two programs CLUSTER/2 (able to build disjoint classifications of unstructured examples) and INDUCE/3 (able to build weakly intersecting classifications of structured examples). The process has three parts: building the structural template, using it to generate attribute-based examples from structured examples, and applying CLUSTER/2 to generate a classification.

7.2.1. Building the structural template

A structural template T is a VL_2 complex that is maximally specific and covers all examples. Fig. 7.1 shows three blocks-world figures e_1, e_4, e_7 and their structural template T. The template figure is composed of a top part that is not a circle and another part beneath it that is curved, medium sized, and shaded. The template description (which is a characteristic description of all examples) is generated by forming the structured REFUNION of all examples.

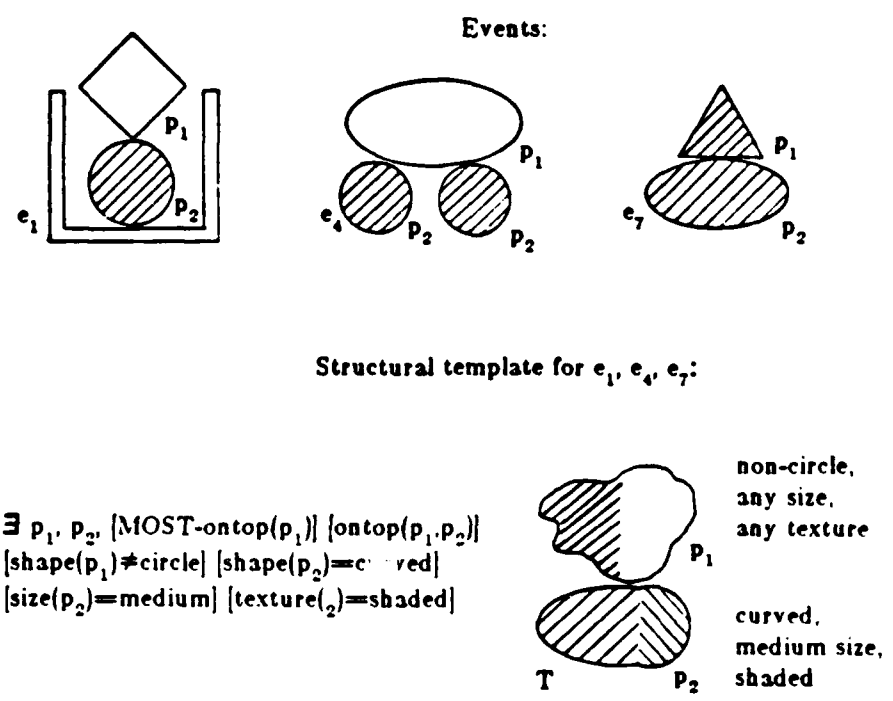


Figure 7.1. A structural template for three blocks-world figures.

The algorithm for structured REFUNION is presented in Sec. 6.5.2.

7.2.2. Deriving attribute-based descriptions for structured examples

Given structural template T and a set of examples E, the process of structural template matching (Sec. 6.4) is used to generate a set of attributes x_1, x_2, ..., x_s and an attribute based description for each example in E. Some examples may give rise to more than one attribute based description if there are multiple ways to match the template to the example. The derived attribute-based descriptions describe only the parts of the structured examples covered by the template structure. Additional steps are taken to add descriptors to quantitatively describe the residual structures not covered by the structural template.

An inventory is made to generate descriptions of parts not covered by the structural template. For the three example figures in Fig. 7.1, the "u"-shape part of figure e_1 is not covered by the template. All such structure fragments are collected and separately classified by recursive application of the conceptual clustering method. When that is completed, each class of structure fragments is tagged with its class number. In the example (which is trivial with respect to building a classification of structure fragments) the "u"-shape fragment would be classified into fragment class 1.

For each connection[2] of fragments to a part covered by the structural template, an attribute is created called "number of fragment i types connected to part j" or more briefly "#iatj" where i is the fragment class number and j is an ordinal that uniquely identifies a part in the structural template. Values for the derived fragment connection attributes are calculated and appended to the attribute descriptions of the examples. In the case of the illustration involving the examples in Fig. 7.1, the ordinal for the second part in the template is 2 and the fragment connection attribute is denoted "#1at2" which stands for "the number of class-1 structure fragments connected at template part 2." The selectors [#1at2=1], [#1at2=0], [#1at2=0] are appended to the attribute-based descriptions for e_1, e_4, and e_7, respectively.

Other ways exist to add attributes to describe the structure not covered by the template. A set of *conditional* attributes may be generated whose value sets contain the two additional values *knowable* and *unknowable*. Such an attribute could be generated for a selector that contains a function whose argument denotes a part which is not covered by the structural template and which is not present in all examples. The value set of a conditional attribute is arranged into a hierarchy in which the values (or hierarchy structure) for the underlying function domain are all subordinate to the generalized value *knowable*. The value *unknowable* is also added, yielding a value hierarchy similar to the one shown in Fig. 7.2 that was built from

2 Connection is defined in terms of graph structure as an existentially quantified variable node for a part covered by the structural template and another for a part not covered by the template both linked to a common selector node

the domain of the function *shape*.

To use a conditional descriptor, a structure template is constructed that fits those examples having the appropriate structure fragment. The process of template matching determines which quantified variable denotes the part described by the conditional descriptor. For examples which do match the structure template, the conditional descriptor takes its value from the corresponding selector node in the graph. If the structure template matches an example in more than one way, independent attribute-based descriptions are generated for each match. For examples which do not match the structure template (because the appropriate structure fragment is not present), the conditional descriptor value *unknowable* is asserted. The automatic construction of conditional descriptors has not been implemented in CLUSTER/S.

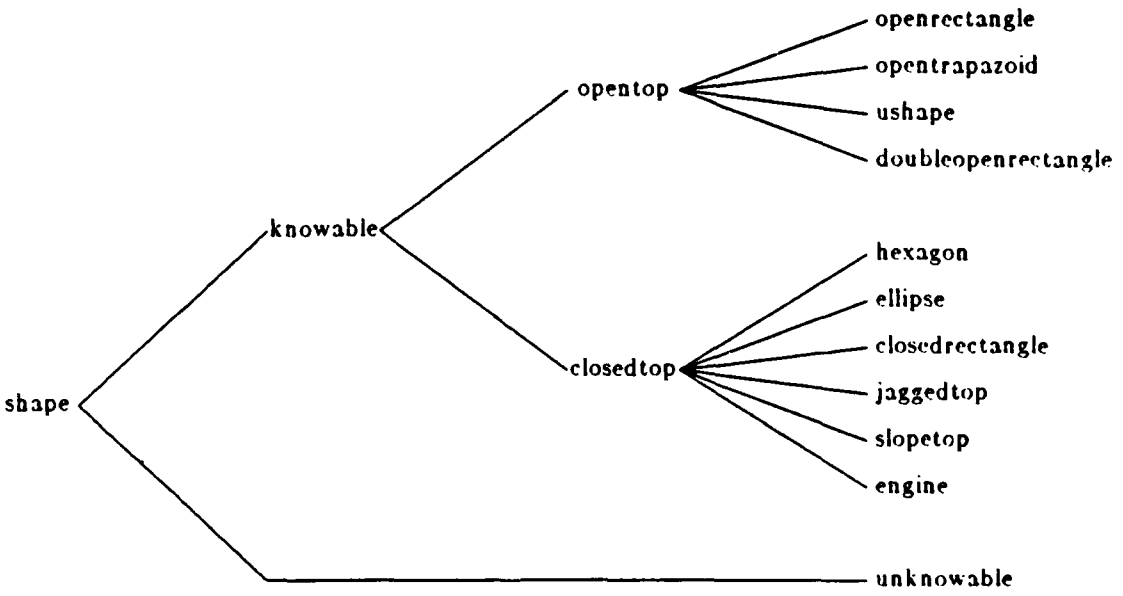


Figure 7.2. An augmented value hierarchy for the conditional descriptor shape.

Presently such descriptors are suggested to the program by the user.

7.2.3. Applying unstructured clustering to derived attributes

The previous section described the method used to derive a set of attribute-based descriptions from structured descriptions in a manner that focuses attention to the parts of each example which are described by the developed structural template. Some descriptive information is included to partially characterize the structure of examples which is not universal. Though not explicitly stated, the zero-place functions which are generated by the application of descriptor construction transformations are propagated into the attribute-based descriptions and are used by the classification building process. When background inference rules are provided, they are processed before the above steps so that selectors they may generate undergo the same structural template processing as selectors that are initially given.

The derived descriptions (that are quantifier-free) are processed by the program CLUSTER/2 as described in Chapter 5. One alteration of the NID procedure used by CLUSTER/2 is necessary to enable it to properly repartition derived examples based on the structured example they represent. This ability is needed only when the template matching finds multiple isomorphisms for an example, giving rise to two different attribute-based representations for the same structured example. NID must take care to arrange for all such multiple representations to be placed into the same class.

7.3. An example problem: Classifying toy trains

Consider a simple example based on rephrasing the problem known as "East- and West-bound trains" [Larson, 1977; Michalski, 1980a] shown in Fig. 7.3. In the original formulation of the problem, given are two collections of trains, those that are "East-bound" (A to E) and those that are "West-bound" (F to J). These trains are highly structured, each consisting of a sequence of cars of different shapes and sizes. The individual cars carry a variable number of loads of different shapes.

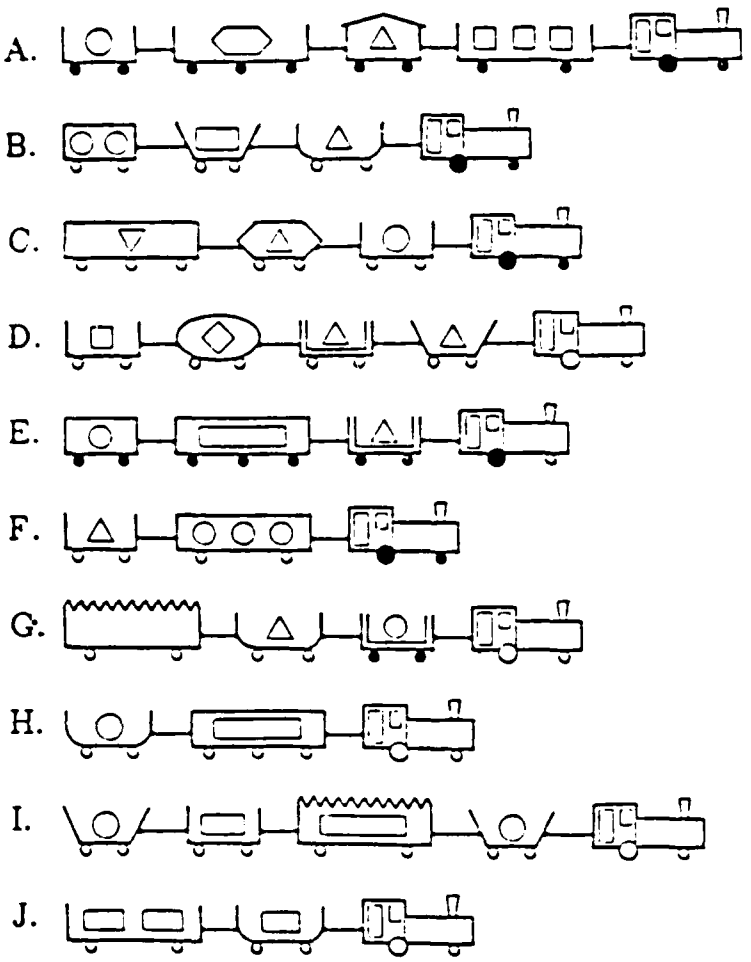


Figure 7.3. How would you classify these trains?

Suppose the class labels are removed (as in Fig. 7.3) and the problem is to create a meaningful classification of the collection of all 10 trains. Research on this problem has investigated both human solutions and those produced by the methodology of this chapter. Sec. 7.3.1 below describes the classifications produced by humans and then the machine-generated solutions are discussed in Sec. 7.3.2.

7.3.1. Human classifications of toy trains

Experiments to generate human classifications of the toy trains were conducted by Medin at the University of Illinois Department of Psychology [Medin, Wattenmaker, and Michalski]. The ten trains (Fig. 7.3) were placed on separate index cards so they could be arranged into groups by the subjects in the experiment. Each subject was instructed to partition the trains into the following groupings and to state the rationale used.

Grouping 1: Arrange the trains into any number of groups of conceptually similar objects.

Grouping 2: Arrange the trains into two equal groups of similar objects.

Grouping 3: Arrange the trains into any number of groups of conceptually similar objects plus an "other" category to hold any unusual or hard to classify trains.

The experiment was completed by 31 subjects who made a total of 93 classification schemes for partitioning the objects. The most popular criterion for classification (17 repetitions) was the number of cars in the trains (an attribute that characterizes each train as a whole and is simple). The three clusters formed were: trains containing 2-cars, 3-cars, and 4-cars, respectively. The second most frequent classification of the trains, based on engine wheel color, occurred 7 times. These classifications are shown in Fig. 7.4. Typically, the classifiers used by the subjects in the experiment to describe the classes of trains were concepts expressed as the conjunction of selected train properties.

Of the 93 classifications produced, 40 of them were unique to one subject in the experiment.[3] Although definite conclusions cannot be drawn from a single experiment, the results suggest that in the absence of explicit goals for a classification, there exists a strong pattern of

3 For a detailed analysis of these results, see [Medin, Wattenmaker and Michalski, 1984]

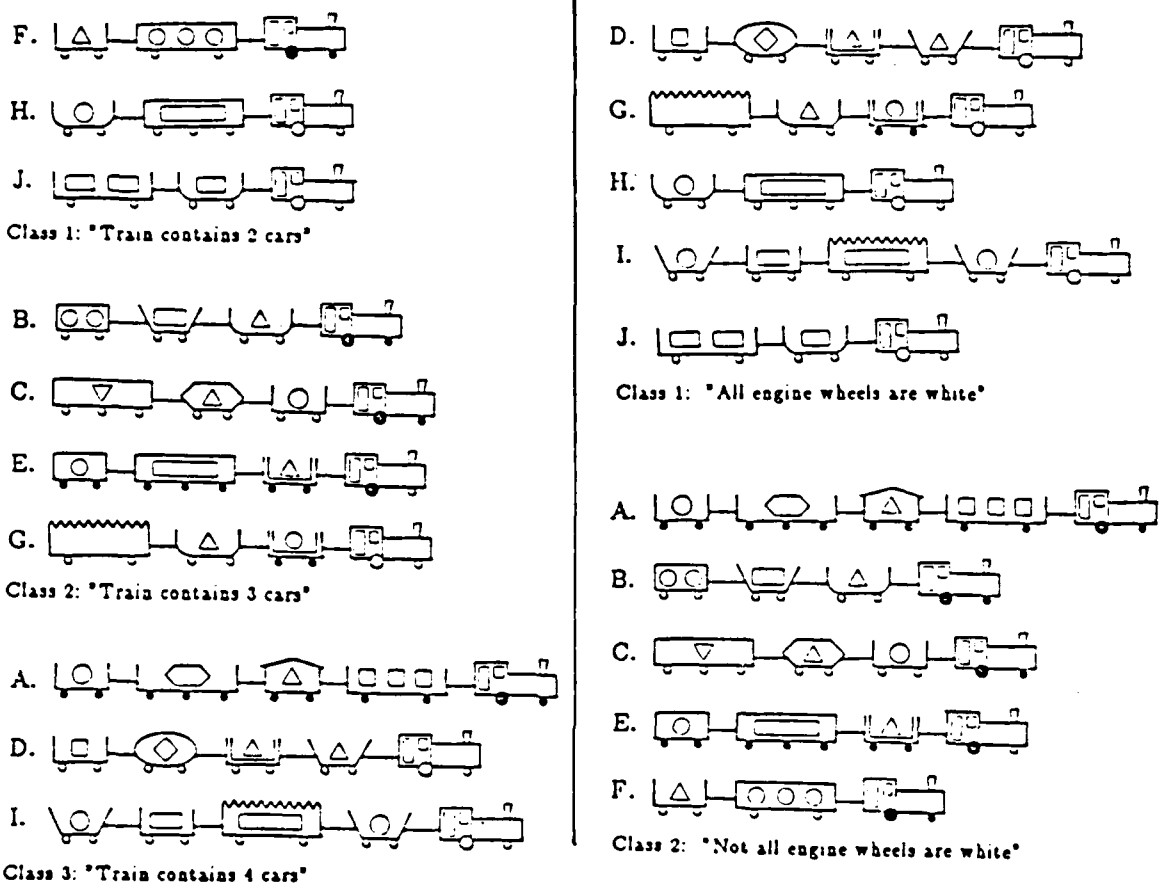


Figure 7.4. The two most popular classifications produced by people.

uniformity along with a wide spectrum of singleton solutions. It is possible that human classifications are generated to satisfy a criterion that is acquired from experiences. Judging from the human classifications, machine classification methods should take into account either explicit goals of the classification, or in the absence of explicit goals, a syntactic criterion that biases the generated descriptions towards simple conjunctive forms that people prefer.

7.3.2. Weaknesses of the east-bound versus west-bound classification

Before presenting the solutions obtained using the method that employs structural template characterization, consider one possible solution based on the discriminant descriptions of East-bound and West-bound trains, as found by Larson using the program INDUCE [Larson, 1977]. These descriptions are:

East-bound trains:

"A train is East-bound if it contains a short, closed car."

West-bound trains:

"A train is West-bound if it contains two cars or if there is a car with a jagged top."

When viewed as a classification involving two classes (East-bound and West-bound), the above class descriptions are atypical of those produced by people because the description of the West-bound trains contains disjunction. By breaking the description of West-bound trains at the disjunction, three classes described by conjunctive concepts are generated:

Class 1: trains that contain a short, closed car;

Class 2: trains that have two cars;

Class 3: trains that contain a jagged-top car.

The above classification is still in a form that is not typical of classifications proposed by a person. The difference is that people tend to describe classes with references to the same attributes (i.e., they tend to propose disjoint class descriptions). In this classification, one class is described in terms of the number of cars, another class is described in terms of the existence of a jagged-top car, and a third class is described in terms of the existence of a short car of a particular shape. The descriptions of the three classes are not disjoint.

In the trains of Fig. 7.3, there is no observed train that contains a short closed car *and* a jagged-top car. If such a train were to exist, it would be described simultaneously by two of the above class descriptions. This indicates that the descriptions are *weakly intersecting* rather than

disjoint. When people are asked to build classifications, they typically form classes with disjoint descriptions, as in the study mentioned above. Thus, if the goal of the clustering process is to generate classifications similar to those produced by people, then the classes should be described by conjunctive concepts that are disjoint. An experiment that does this through the use of a structural template is described below.

7.3.3. Disjoint classifications generated by using a structural template

For this problem, the structured descriptions of each train involve the descriptors *contains, infront, number of cars, number of wheels, wheel type, car shape, car length, cargo shape,* and *number of cargo items carried.* For example, the description of train A is

$\exists$ t_1, car_1, car_2, car_3, car_4, car_5, [contains(t1,car_1)] [contains(t1,car_2)]
[contains(t1,car_3)] [contains(t1,car_4)] [contains(t1,car_5)] [number-of-cars(t1)=5]
[infront(car_1,car_2)] [infront(car_2,car_3)] [infront(car_3,car_4)] [infront(car_4,car_5)]
[number-of-wheels(car_1)=2] [number-of-wheels(car_2)=2]
[number-of-wheels(car_3)=2] [number-of-wheels(car_4)=3]
[number-of-wheels(car_5)=2] [wheel-type(car_1)=black] [wheel-type(car_2)=black]
[wheel-type(car_3)=black] [wheel-type(car_4)=black] [wheel-type(car_5)=black]
[car-length(car_1)=long] [car-length(car_2)=long] [car-length(car_3)=short]
[car-length(car_4)=long] [car-length(car_5)=short] [car-shape(car_1)=engine]
[car-shape(car_2)=open-rectangle] [car-shape(car_3)=slope-top]
[car-shape(car_4)=open-rectangle] [car-shape(car_5)=open-rectangle]
[number-of-items-carried(car_1)=0] [number-of-items-carried(car_2)=3]
[number-of-items-carried(car_3)=1] [number-of-items-carried(car_4)=1]
[number-of-items-carried(car_5)=1] [cargo-shape(car_2)=rectange-load]
[cargo-shape(car_3)=triangle-load] [cargo-shape(car_4)=hexagon-load]

[cargo-shape(car_5)=circle-load].

The structural template for the trains generated by applying the methods presented in this chapter is the description

$$\exists\ car_1,\ car_2,\ car_3,\ [\text{MOST-infront}(car_1)][\text{infront}(car_1,car_2)]$$
$$[\text{infront}(car_2,car_3)][\text{car-shape}(car_1)=\text{engine}]$$

which identifies the engine and the next two cars in each train. The 51 variables derived from the above structural template are listed in Fig. 7.5. Two fragment templates f1 and f2 were

---

car-shape.car2	#cars(car-shape=ushape)
car-shape.car3	#cars(car-shape=doubleopenrectangle)
forall-cars(#wheels=2)	#cars(car-shape=hexagon)
forall-cars(wheel-type=black)	#cars(car-shape=ellipse)
forall-cars(wheel-type=clear)	#cars(length=short)
forall-cars(cargo-shape=rectangle)	#cars(length=long)
length.car2	#cars(cargo-shape=hexagon)
length.car3	#cars(cargo-shape=circle)
cargo-shape.car2	#cars(cargo-shape=rectangle)
cargo-shape.car3	#cars(cargo-shape=triangle)
#cars	#cars(#cargo=0)
#cargo.car2	#cars(#cargo=1)
#cargo.car3	#cars(#cargo=2)
#wheels.car2	#cars(#cargo=3)
SAME-#wheels.car1.car2.car3	#cars(#wheels=2)
SAME-wheel-type.car1.car2.car3	#cars(#wheels=3)
wheel-type.car1	#cars(wheel-type=mixed)
wheel-type.car2	#cars(wheel-type=clear)
wheel-type.car3	#cars(wheel-type=black)
#cars(car-shape=closedrectangle)	#different-car-shapes
#cars(car-shape=jaggedtop)	#different-cargo-shapes
#cars(car-shape=slopingtop)	#different-#cargo
#cars(car-shape=opentop)	#different-#wheels
#cars(car-shape=closedtop)	#different-wheel-type
#cars(car-shape=openrectangle)	LEAST-infront.car3
#cars(car-shape=opentrapazoid)	

Figure 7.5. Attributes of trains derived from program-generated structural template.

---

used. They are

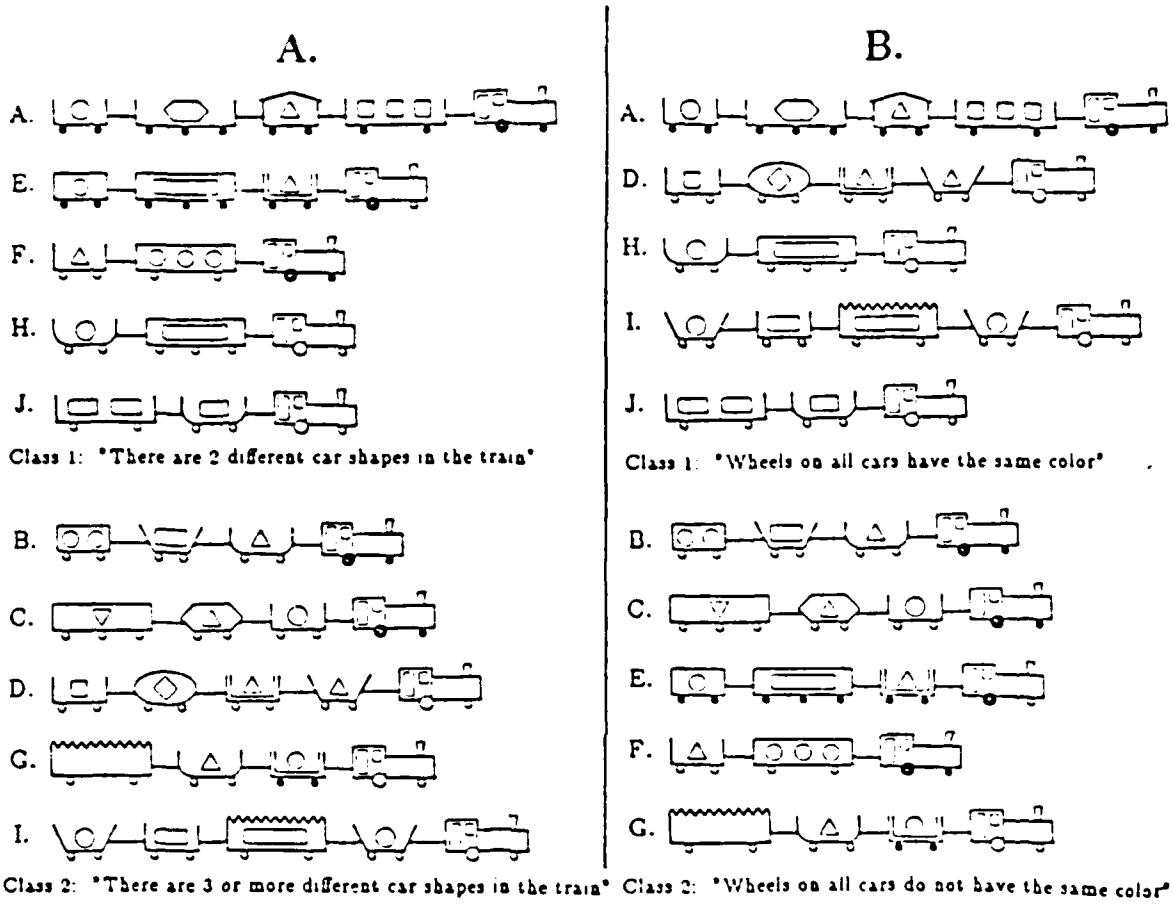
f1: $\exists$ car_1, car_2, car_3, car_4, [MOST-infront(car_1)] [infront(car_1,car_2)] [infront(car_2,car_3)] [infront(car_3,car_4)]

f2: $\exists$ car_1, car_2, car_3, car_4, car_5, [MOST-infront(car_1)] [infront(car_1,car_2)] [infront(car_2,car_3)] [infront(car_3,car_4)] [infront(car_4,car_5)]

These fragment templates were used to give rise to the connected structure attributes #1at3, #2at3, and the conditional descriptor attributes car-shape.car4, cargo-shape.car4, #cargo.car4, #wheels.car4, and wheel-type.car4. The domains of each of the conditional descriptor attributes was augmented with *knowable* and *unknowable* values as described above. With these 7 fragment attributes added to the previous 51 attributes, each train is described by values for 58 derived attributes.

The generated attribute descriptions were processed using a classification evaluation criterion LEF that minimizes the number of attributes used in a description (*minimum commonality*), and maximizes the number of attributes that singly discriminate between all classes (*maximum dimensionality*). Minimizing commonality tends to conflict with the other criterion. This was handled by specifying a high tolerance (95%) for the first criterion and zero tolerances for the second one. The LEF is set up to apply the first criterion to filter out the 5% most specific solutions (measured terms of number of selectors) and then to select a solution with the greatest number of discriminant attributes.

Classification A. shown in Fig. 7.6 was generated by the program with two different class descriptions. The top class ("there are 2 different car shapes in the train") was also described as "the third car from the engine (if it exists) has black wheels." The bottom class ("there are more than 2 different car shapes in the train") was also described as "the third car from the engine exists and has white wheels." Classification B. in Fig. 7.6 is based on the derived predicate *SAME-wheel-type*. Both classifications received the same evaluation criterion score and were



Criterion LEF: minimize commonality with 95% tolerance, then maximize dimensionality.

Figure 7.6. Two sample classifications found by structural template characterization.

considered to be alternative classifications. Solutions of the kind shown in Fig. 7.6 are appealing because the class descriptions are disjoint and the difference between classes is striking, yet not obvious by casual inspection.

7.3.4. A classification using an additional background inference rule

It is possible to guide classification building through explicit inference rules supplied as background knowledge. Suppose that the knowledge base has been augmented to include an inference rule to identify trains carrying toxic chemicals. This section presents a classification generated using attributes derived by the application of such an inference rule.

Using the illustrations of the trains, a toxic chemical container is identified as a single sphere (circle) riding in an open-top car. The logical inference rule (ℓ-rule) supplied to the program is

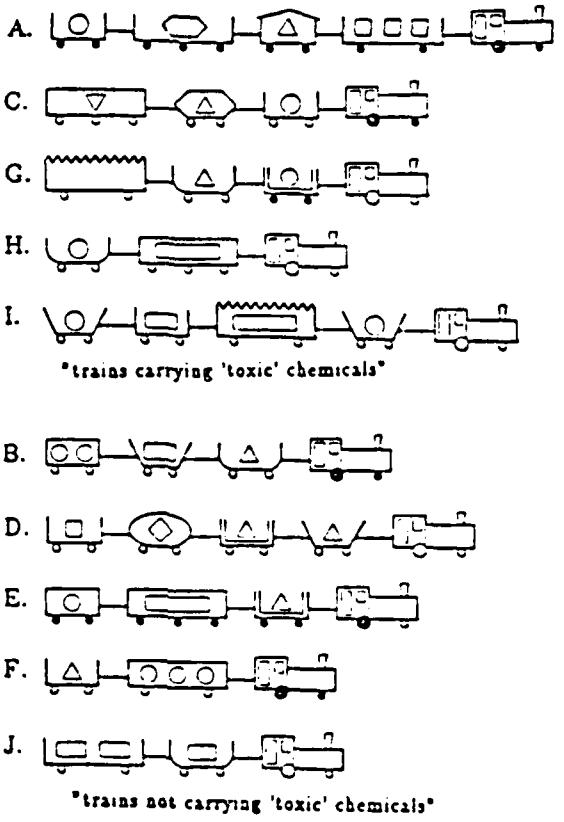
[contains(train,car)][car-shape(car)=opentop]

[cargo-shape(car)=circle][number-of-items-carried(car)=1] $\Longleftrightarrow$ [toxic-chemicals(train)]

In the above inference rule, equivalence is used to indicate that the negation of the condition part is sufficient to assert the negative of the consequence. After the application of this rule, all trains will have descriptions containing either the toxic-chemicals predicate or its negation. The structural template generated by INDUCE/3 will now contain the additional predicate *toxic-chemicals(train)*. Other relevant inferences that bear on this problem are:

- toxic chemicals are dangerous,
- dangerous things are important,
- important things should have high selection value (high preference score).

The classification produced in this case is shown in Fig. 7.7. The classification criterion used to generate this classification differs slightly from the criterion used for the results in Fig. 7.6. Instead of minimizing commonality, the first criterion applied is maximizing the preference of the descriptions and the derived attribute "toxic-chemicals(train)" is given a large preference score (as dictated by the above inference chain) to reflect its importance in the resulting descriptions.



Criterion LEF: maximize similarity with 90% tolerance, then maximize dimensionality.

Figure 7.7. A classification produced using the 'toxic chemicals' inference rule.

CHAPTER 8

CONCLUSION

8.1. Summary

Interesting conceptual patterns in data have been discovered by automated conceptual data analysis techniques presented in this thesis that *learn from observation*. Such learning mechanisms play important roles in knowledge acquisition systems used for problem solving and for building knowledge bases for expert systems. Three techniques based on the methodology of *conjunctive conceptual clustering* were presented in Chapters 5, 6, and 7. The underlying methodology for these conceptual clustering methods (presented in Chapters 3 and 4) is based on both the initial ideas of Michalski [1980b] and the collaborative ideas of Michalski and Stepp [1983].

The numerically based techniques of feature extraction, multi-dimensional scaling, and numerical taxonomy perform data analysis functions without the ability to provide any conceptual interpretations. By using the inductive inference technique of *learning from examples* in a fashion patterned after *dynamic clustering*, a general methodology for conceptual clustering has been produced. This methodology proposes that objects be grouped into classes not on the basis of a mathematical measure of similarity but on the basis of *conceptual cohesiveness* that takes into account the similarity between objects and also surrounding objects and concepts for describing collections of objects.

Since there are neither "right" nor "wrong" solutions to conceptual clustering or classification building problems, some goal and/or bias of the clustering process is necessary to guide the problem solution towards the kinds of solutions which are useful or interesting. Such

guidance can come from three sources: the explicit goal of the process expressed using a LEF (the classification goal), the application of background knowledge to produce derived goals, and the built-in biases of the algorithm.

Classification construction problems occur when one wants to organize and classify observations according to underlying conceptual patterns. Problems of this type include classifying physical or chemical structures, analyzing genetic sequences, building taxonomies of plants or animals, characterizing visual scenes, etc. In different situations an adequate representation may be attribute based (if objects do not have structure) or it may be predicate/function based (if objects have structure). Problems with the former type of representation are handled by the method presented in Chapter 5 for clustering unstructured objects. Problems with the latter type of representation are handled by the methods presented in Chapters 6 and 7 for clustering structured objects.

The approach taken for generating classifications has the advantage of depending mostly on the descriptions of the individual events (especially the differences between seed events) to suggest the concepts that may be relevant for describing classes of events as a whole. This approach is hard-biased (i.e., biased by built-in hard-coded algorithms) by the representation language (which is quite general) and the available generalization transformations (which are also quite general). The user can add as much additional soft-bias (i.e., bias by problem statement and design) as desired, based on the definition of the LEF, the definition of function domains and value hierarchies, and supplied background knowledge inference rules.

The presented methods are domain independent and have been applied to a wide variety of problems. The example in Chapter 5 was an application of the method to the problem of discovering classes of soybean disease. In that problem, cases of diseased soybean plants each described by 35 attributes were grouped into four classes. The classes and their descriptions corresponded to four diseases recognized by plant pathologists.

In another experiment performed by Spackman [Michalski, Baskin, Spackman, 1982], the method was used to build a classification of patients with various craniosynostosis syndromes. The two-class solution consisted of one group of patients having one syndrome and another group containing a mixture of patients having one or the other of two syndromes.

The presented conceptual clustering method has also been used to build classifications of 12 microcomputers [Michalski and Stepp, 1983] and a collection of 100 Spanish folksongs [Stepp, 1980].

8.2. Areas of further research

In solving a problem, it is important to distinguish between constraints that are part of the problem and those that are imposed by the process of obtaining the solution. The presented methodology for building classifications through conceptual clustering has inherent constraints and inherent advantages. The discussion will include analysis of the types of classifications which can be built, model-driven versus data-driven approaches, the application of background knowledge, and the built-in (hard) biases of the algorithms.

8.2.1. Types of classifications

This thesis has presented classification building by the application of conjunctive conceptual clustering techniques. The hierarchical classifications which are produced have classes that are disjoint and class descriptions (complexes) that are disjoint or only weakly intersecting. These kinds of classifications correspond to the kinds often produced by people when asked to create a classification scheme for a collection of objects [Medin, Wattenmaker, Michalski, 1984]. A classification with disjoint classes is not the only kind of classification which people build, however. Other kinds can be considered.

Interesting patterns in collections of objects may involve objects belonging to more than one class at the same time. For example, when classifying athletes, the classes of *basketball players* and *baseball players* may be good ways to describe the collection of athletes, but these

two classes may not be disjoint. When the requirement for disjointness is removed, the goal of the classification could continue to be optimizing the *simplicity* and *fit*, i.e., trying to find classes that fit the data well (being specific enough) while also being simple (being general enough). The algorithms presented in Chapters 5, 6, and 7 can generate classifications containing non-disjoint classes by removing the NID procedure that transforms non-disjoint complexes into disjoint ones.

8.2.2. Model-driven versus data-driven methods

The methodology for conceptual clustering is a mixture of model-driven and data-driven approaches. The data (event descriptions) are used to select seed events and to drive the *star generation* process, constructing hypothetical class descriptions that are processed as necessary by NID to make them disjoint. Once the class descriptions (models) are synthesized via star generation, they are evaluated according to the LEF and the best ones are reported as final results. *Overall, the described algorithms are more* data-driven than they are model-driven.

An approach that is more model-driven is certainly possible. If simplicity of class descriptions is an element of the classification goal, a model-driven approach could select individual descriptors and use them to build several candidate class descriptions, generated in order by increasing length. It would try a number of descriptions containing 1 selector; then some with 2 selectors; etc. The algorithm would begin by selecting an initial descriptor whose value set can be partitioned into k subsets. This descriptor would be used in k class descriptions, each using a different value subset (thus making the descriptions disjoint). Next, the events would be partitioned to match the descriptions and the entire clustering would be evaluated with respect to the LEF. The process would be repeated using different descriptors and different numbers of descriptors. The model which scores the best would be the final result.

The above "model-driven" approach depends on heuristics for combining descriptors and for knowing when to halt. Although experimentation has not been conducted on this type of

approach, it is reasonable to expect the heuristics to be very important and for the generated classifiers to reflect the imposed heuristic control. Thus, the classifications would tend to match the models which the heuristics promote, rather than underlying patterns of conjunctive concepts in the data. This approach would be good when building a classification to fit into existing theory, while the "data-driven" approach as presented in Chapters 5, 6, and 7 is better for discovering interesting new or unexpected conjunctive concepts in the data.

8.2.3. Application of background knowledge

The application of background knowledge to problems of unsupervised learning is highly beneficial. It can permit the construction of relevant descriptors from descriptors given initially that may be rather irrelevant by themselves to the conceptual patterns in the data. In the approach outlined in Chapter 6, background knowledge about descriptor domains is used to guide generalization transformations and background knowledge in the form of background inference rules is applied to events before the actual clustering process begins. The background inference rules are used to augment descriptions of the events by adding new selectors. Then, the clustering process is performed using augmented event descriptions.

An alternate way to perform clustering in problems with background inference rules would be to adopt the model-driven approach above and to use a *backward-chaining* control scheme to work backwards through the background rule base from proposed model solutions to event descriptions and/or generalized event descriptions. The candidate classification schemes produced would then be evaluated via a LEF, with the best classification preserved as the final result.

8.2.4. Built-in biases of the algorithms

The most fundamental built-in biases (*hard* bias) of the algorithms are the constraints imposed by the choice of representation language. This choice affects the composition of concepts and the kinds of applicable generalizing and specializing transformations which can be

applied to events.

Although conjunctive concepts are a natural component of human expression, they are by no means the only descriptive form used. The advantage of the conjunctive concept is its expressiveness coupled with its ease of understanding (especially with extensions to predicate logic such as internal disjunction) and ease of transformation by a variety of inference rules. The constraint of conjunctive-only concept forms could be removed by introducing other logical operators (e.g., disjunction, equivalence, implication). This would invite the introduction of additional inference rules for applying generalizing and specializing transformations for descriptions involving the added operators. The language could also be extended with statistical and probabilistic expressions (e.g., relations on fuzzy sets [Zadeh, 1965]).

Within the constraint of conjunctive form concepts, there are additional generalizing and specializing transformations that could be applied. To consider one such change, consider an adaptation of the *closing the interval rule*. This transformation generalizes the value set of a linear-domain function until it contains all values in an enclosing interval of values. It is an arbitrary bias of the algorithm to insist that the generalization be one interval, rather than many intervals. An alternative approach is to determine the number of intervals to generate by using a threshold limit on the number of missing values that are filled in to complete an interval. The intervals would be arranged in such a way that the given observed values are covered with a minimum number of intervals that collectively do not exceed the threshold limit on the number of "filled-in" values.

8.3. Promising future applications

There are two interesting areas of future research where the presented methods of conjunctive conceptual clustering can be applied when they are made more robust. The first area of application is self-organizing (or self-reorganizing) knowledge-based systems. A system can collect data about its own patterns of access to the knowledge store. Conceptual patterns in

this data can be used to help structure the knowledge layout in ways that accelerate access. As intelligent systems demand ever-increasing amounts of knowledge content, our ability to manually structure the knowledge for efficient use shrinks in relationship to the size of the knowledge base. Worse yet, the patterns of use of the knowledge are not constant, so periodic reevaluation of the knowledge base layout is beneficial.

The second area is a variation of the first. Instead of operating on self-maintained access data, let the conceptual analysis tools operate on the knowledge itself, directed in part by the background knowledge in the database. Important conceptual regularities in the knowledge base may be revealed by this process and it could help automate the scientific inquiry process. Just as one mark of an intelligent system is an ability to reason, another aspect of intelligence is an ability *to observe* and *to form classifications of the observations* based on available background knowledge in a current world model. The work described in this thesis begins a course of research towards finding algorithms to be used to construct such systems.

APPENDIX A

REDISCOVERING CLASSES OF SOYBEAN DISEASE

A.1. Problem description

Given are 47 cases of soybean disease. The cases come from four disease categories recognized by plant pathologists: Diaporthe Stem Canker (events 1 to 10), Charcoal Rot (events 11 to 20), Rhizoctonia Root Rot (events 21 to 30), and Phytophthora Rot (events 31 to 47). These categories were hidden from the clustering program. Each case of soybean disease is described by 35 attribute/value pairs. The attributes used in the experiment and their domains are given below. In the listing of input data which follows, the values are coded as integers, according to the mappings noted in the description of each attribute.

1. time_of_occurrence (linear) 7 values: 0:april 1:may 2:june 3:july 4:august 5:september 6:october
2. plant_stand (nominal) 2 values: 0:normal 1:less_than_normal
3. precipitation (linear) 3 values 0:below_normal 1:normal 2:above_normal
4. temperature (linear) 3 values: 0:below_normal 1:normal 2:above_normal
5. occurrence_of_hail (nominal) 2 values: 0:no 1:yes
6. number_years_crop_repeated (linear) 4 values: 0:none 1:one 2:three 3:four_or_more)
7. damaged_area (nominal) 4 values: 0:scattered_areas 1:low_areas 2:upland_areas 3:whole_fields

8. severity (linear) 3 values: 0:minor 1:potentially_severe 2:severe)

9. seed_treatment (nominal) 3 values: 0:none 1:fungicide :2other

10. seed_germination (linear) 3 values: 0:90%-100% 1.80%-89% 2:less_than_80%

11. plant_height (nominal) 2 values: 0:normal 1:abnormal

12. leaf_condition (nominal) 2 values: 0:normal 1:abnormal

13. leaf_spots–halos (nominal) 3 values: 0:absent 1:with_yellow_halos
    2:without_yellow_halos

14. leaf_spots–margin (nominal) 3 values: 0:water soaked 1:not_water soaked
    2:does_not_apply

15. size_of_leaf_spots (nominal) 3 values: 0:less_than_one_eighth_inch
    1:greater_than_eighth_inch 2:does_not_apply

16. shot_holing (nominal) 2 values: 0:absent 1:present

17. leaf_malformation (nominal) 2 values: 0:absent 1:present

18. leaf_mildew_growth (nominal) 3 values: 0:absent 1:upper_leaf_surface
    2:lower_leaf_surface

19. condition_of_stem (nominal) 2 values: 0:normal 1:abnormal

20. stem_lodging (nominal) 2 values: 0:absent 1:present

21. stem_cankers (nominal) 4 values: 0:absent 1:below_soil 2:slightly_above_soil_line
    3:above_second_node

22. canker_lesion_color (nominal) 4 values: 0:does_not_apply 1:brown
    2:dark_brown_or_black 3:tan

23. fruiting_bodies_on_stem (nominal) 2 values: 0:absent 1:present

24. outer_stem_decay (nominal) 3 values: 0:absent 1:firm_and_dry 2:watery_and_soft

25. mycelium_on_stem (nominal) 2 values: 0:absent 1:present

26. internal_discoloration_of_stem (nominal) 3 values: 0:none 1:brown 2:black

27. sclerotia–internal_or_external (nominal) 2 values: 0:absent 1:present

28. fruit_pod_condition (nominal) 4 values: 0:normal 1:diseased 2:few_or_none 3:does_not_apply

29. fruit_spots (nominal) 5 values: 0:absent 1:colored_spots 2:brown_spots_black_specks 3:distorted pods 4:does_not_apply

30. seed_condition (nominal) 2 values: 0:normal 1:abnormal

31. seed_mold_growth (nominal) 2 values: 0:absent 1:present

32. seed_discoloration (nominal) 2 values: 0:absent 1:present

33. seed_size (nominal) 2 values: 0:normal 1:smaller_than_normal

34. seed_shriveling (nominal) 2 values: 0:absent 1:present

35. root_condition (nominal) 3 values: 0:normal 1:rotted 2:galls_or_cysts

A.2. Input data file

```
  1 "SOYBEAN EXAMPLE D1, D2, D3, D4"

 PARAMETERS
    TRACE  K H1 H2 H3 BASE PROBE CRITERION NIDSPEED
     ON    4  2  1  2  2    2     F           SLOW

F-CRITERION
   #  CRITERION  TOL
   1   SPARS     0.0

 VARIABLES
  #  TYPE  LEVELS
  1     L      7
  2     N      2
  3     L      3
  4     L      3
  5     L      2
  6     L      4
```

This specifies the title of the example

This table specifies that 4 classes are to be produced (k=4) and gives settings for the stopping criterion (base=2, probe=2) and certain depth limits for the PRO search (h1=2, h2=1, h3=2) The nidspeed=slow setting uses the regular NID algorithm. Criterion F is defined below

This table specifies the LEF to be used. The only elementary criterion used is sparseness.

This table gives the type of each variable (attribute) as N-nominal or L-linear and specifies the number of ele-

ments (levels) in the domain

7	N	4
8	L	3
9	N	3
10	L	3
11	N	2
12	N	2
13	N	3
14	N	3
15	N	3
16	N	2
17	N	2
18	N	3
19	N	2
20	N	2
21	N	4
22	N	4
23	N	2
24	N	3
25	N	2
26	N	3
27	N	2
28	N	4
29	N	5
30	N	2
31	N	2
32	N	2
33	N	2
34	N	2
35	N	3

EVENTS

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
6	0	2	1	0	1	0	1	0	2	1	1
6	0	2	1	0	3	0	1	1	1	1	1
3	0	2	1	0	1	0	2	1	2	1	1
3	0	2	1	0	2	1	1	0	1	1	1
5	0	2	1	0	2	0	1	1	0	1	1
4	0	2	1	0	3	0	2	0	2	1	1
6	0	2	1	0	1	1	1	0	0	1	1
3	0	2	1	0	2	0	2	1	1	1	1
5	0	2	1	0	3	1	1	1	2	1	1
4	0	2	1	1	1	0	1	0	2	1	1
5	0	0	2	1	3	3	1	1	2	1	1
6	0	0	2	0	1	3	1	1	0	1	1
5	0	0	2	1	2	2	1	0	2	1	1
3	0	0	1	0	1	2	1	0	0	1	1
4	0	0	1	1	1	3	1	1	1	1	1
3	0	0	2	1	0	2	1	0	1	1	1
6	0	0	1	1	3	3	1	1	0	1	1
5	0	0	2	0	3	2	1	0	2	1	1

The events table defines each case of diseased soybeans. The table is presented in three parts, one with attributes X1 to X12, one with attributes X13 to X24, and one with attributes X25 to X35

4	0	0	1	0	2	3	1	1	1	1	1
6	0	0	2	1	0	2	1	0	0	1	1
0	1	2	0	0	0	1	1	1	2	1	0
2	1	2	0	0	2	1	1	0	2	1	0
2	1	2	0	0	3	1	2	0	1	1	0
0	1	2	0	0	1	1	1	1	1	1	0
0	1	2	0	0	1	1	2	1	2	1	0
3	0	2	0	1	3	1	2	0	1	1	0
0	1	2	0	0	0	1	1	0	1	1	0
2	1	2	0	0	3	1	2	0	2	1	0
4	0	2	0	1	0	1	2	0	2	1	1
0	1	2	0	0	2	1	1	1	1	1	0
3	1	2	0	0	2	1	2	1	1	1	1
2	1	2	0	0	1	1	2	0	0	1	1
0	1	2	1	0	3	1	1	0	0	1	1
1	1	2	1	0	0	1	2	1	1	1	1
1	1	2	0	0	3	1	1	1	2	1	1
2	1	2	1	1	1	1	2	0	2	1	1
3	1	2	0	0	1	1	2	1	0	1	1
2	1	2	1	1	3	1	2	1	2	1	1
0	1	1	1	0	1	1	1	0	0	1	1
0	1	2	1	0	3	1	1	0	2	1	1
0	1	1	1	1	2	1	2	1	0	1	1
2	1	1	0	0	3	1	2	0	2	1	1
1	1	2	1	1	2	3	1	1	1	1	1
1	1	2	0	0	0	1	2	1	0	1	1
1	1	2	1	1	3	1	2	0	1	1	1
0	1	2	1	1	1	1	1	0	0	1	1
3	1	1	0	0	2	1	2	1	2	1	1

EVENTS

X13	X14	X15	X16	X17	X18	X19	X20	X21	X22	X23	X24
0	2	2	0	0	0	1	0	3	1	1	1
0	2	2	0	0	0	1	0	3	1	1	1
0	2	2	0	0	0	1	0	3	0	1	1
0	2	2	0	0	0	1	1	3	0	1	1
0	2	2	0	0	0	1	1	3	1	1	1
0	2	2	0	0	0	1	0	3	1	1	1
0	2	2	0	0	0	1	1	3	1	1	1
0	2	2	0	0	0	1	0	3	0	1	1
0	2	2	0	0	0	1	1	3	0	1	1
0	2	2	0	0	0	1	0	3	1	1	1
0	2	2	0	0	0	1	0	0	3	0	0
0	2	2	0	0	0	1	0	0	3	0	0
0	2	2	0	0	0	1	1	0	3	0	0
0	2	2	0	0	0	1	0	0	3	0	0
0	2	2	0	0	0	1	1	0	3	0	0
0	2	2	0	0	0	1	0	0	3	0	0
0	2	2	0	0	0	1	0	0	3	0	0
0	2	2	0	0	0	1	0	0	3	0	0

0	2	2	0	0	0	1	0	0	3	0	0
0	2	2	0	0	0	1	1	0	3	0	0
0	2	2	0	0	0	1	0	1	1	0	1
0	2	2	0	0	0	1	0	1	1	0	1
0	2	2	0	0	0	1	0	1	1	0	1
0	2	2	0	0	0	1	0	1	1	0	1
0	2	2	0	0	0	1	0	1	1	0	1
0	2	2	0	0	0	1	1	1	1	0	1
0	2	2	0	0	0	1	0	1	1	0	1
0	2	2	0	0	0	1	0	1	1	0	1
0	2	2	0	0	0	1	1	1	1	0	1
0	2	2	0	0	0	1	0	1	1	0	1
0	2	2	0	0	0	1	0	2	2	0	0
0	2	2	0	0	0	1	0	1	2	0	0
0	2	2	0	0	0	1	0	1	2	0	0
0	2	2	0	0	0	1	0	2	2	0	0
0	2	2	0	0	0	1	0	2	2	0	0
0	2	2	0	0	0	1	0	1	2	0	1
0	2	2	0	0	0	1	0	2	2	0	0
0	2	2	0	0	0	1	0	2	2	0	1
0	2	2	0	0	0	1	0	1	2	0	0
0	2	2	0	0	0	1	0	1	2	0	0
0	2	2	0	0	0	1	1	2	2	0	1
0	2	2	0	0	0	1	0	1	2	0	0
0	2	2	0	0	0	1	0	2	2	0	1
0	2	2	0	0	0	1	0	2	2	0	0
0	2	2	0	0	0	1	1	1	2	0	1
0	2	2	0	0	0	1	0	1	2	0	1
0	2	2	0	0	0	1	0	2	2	0	0

EVENTS

X25	X26	X27	X28	X29	X30	X31	X32	X33	X34	X35
0	0	0	0	4	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0	0	0
0	2	1	0	4	0	0	0	0	0	0
0	2	1	0	4	0	0	0	0	0	0
0	2	1	0	4	0	0	0	0	0	0
0	2	1	0	4	0	0	0	0	0	0
0	2	1	0	4	0	0	0	0	0	0
0	2	1	0	4	0	0	0	0	0	0
0	2	1	0	4	0	0	0	0	0	0
0	2	1	0	4	0	0	0	0	0	0
0	2	1	0	4	0	0	0	0	0	0

```
0  2  1  0  4  0  0  0  0  0  0
0  0  0  3  4  0  0  0  0  0  0
1  0  0  3  4  0  0  0  0  0  0
0  0  0  3  4  0  0  0  0  0  0
1  0  0  3  4  0  0  0  0  0  0
0  0  0  3  4  0  0  0  0  0  0
1  0  0  3  4  0  0  0  0  0  0
0  0  0  3  4  0  0  0  0  0  1
1  0  0  3  4  0  0  0  0  0  0
1  0  0  3  4  0  0  0  0  0  0
0  0  0  3  4  0  0  0  0  0  0
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
0  0  0  3  4  0  0  0  0  0  1
```

A.3. Generated output

The output generated from the above input data is shown below. Although the goal of this research is not strictly directed at reproducing human classifications of events, comparisons of machine and human classifications is an interesting part of the work. In this example, events 1 to 10, 11 to 20, 21 to 30, and 31 to 47 were classified as the same soybean disease by human plant pathologists. In the machine classification, these same categories are reproduced.

```
 CONJUNCTIVE CONCEPTUAL CLUSTERING PROGRAM CLUSTER/2   LAST UPGRADE  04/10/84

PARAMETERS
MINK MAXK TRACE H1 H2 H3 INITMETHOD MIDSPEED COVERTYPE CRITERION BASE PROBE BETA MAXHEIGHT MINSIZE
   2   4    ON   2  1  2   RANDOM     SLOW  HIERARCHIAL F          2     2   3 0      2       4

F-CRITERION
  #   CRITERION  TOLERANCE
  1    SPAR       0 00

                    SOYBEAN EXAMPLE FROM CHAPTER 5
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1  K=2, CRITERION=F                          The classes in the first level of
```

```
INTERMEDIATE RESULTS
ITER/CLASS# VL-RULE                                                    SEED --COSTS---
                                                                              SPAR
1     1  [X7=0,1,2][X8=1  2][X9=0][X11=1][X13=0][X14=2]                  1    1 1E+008
         [X15=2][X16=0][X17=0][X18=0][X19=1][X21=0,1,3]
         [X24=0,1][X26=0,2][X28=0,3][X29=4][X30=0]
         [X31=0][X32=0][X33=0][X34=0][X35=0,1]
         EVENTS COVERED  1,4,6,7,10,13,14,16,18,20,22,23,26,27,28,29,32,
                         33,36,39,40,42,45,46
1     2  [X7=0,1,3][X8=1  2][X9=1][X11=1][X13=0][X14=2]                  2    1 5E+008
         [X15=2][X16=0][X17=0][X18=0][X19=1][X24=0,1]
         [X26=0,2][X28=0,3][X29=4][X30=0][X31=0][X32=0]
         [X33=0][X34=0][X35=0,1]
         EVENTS COVERED  2,3,5,8,9,11,12,15,17,19,21,24,25,30,31,34,35,
                         37,38,41,43,44,47
1   TOTALS                                                                    2 6E+008
    (1221 MS)

2     1  [X3=2][X4=0  1][X7=0,1][X8=1  2][X9=0,1][X11=1]                 23    1 4E+006
         [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]
         [X19=1][X21=1,3][X22=0,1][X24=1][X26=0][X27=0]
         [X28=0,3][X29=4][X30=0][X31=0][X32=0][X33=0]
         [X34=0][X35=0,1]
         EVENTS COVERED  1,2,3,4,5,6,7,8,9,10,21,22,23,24,25,26,27,28,29,30
2     2  [X7=1,2,3][X8=1  2][X9=0,1][X11=1][X12=1]                       31    1 4E+007
         [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]
         [X19=1][X21=0,1,2][X22=2,3][X23=0][X24=0,1]
         [X25=0][X26=0,2][X28=0,3][X29=4][X30=0][X31=0]
         [X32=0][X33=0][X34=0][X35=0,1]
         EVENTS COVERED  11,12,13,14,15,16,17,18,19,20,31,32,33,34,35,
                         36,37,38,39,40,41,42,43,44,45,46,47
2   TOTALS                                                                    1 5E+007
    (2665 MS)

3     1  [X1=3  6][X2=0][X4=1  2][X8=1  2][X9=0,1]                         6    1 8E+006
         [X11=1][X12=1][X13=0][X14=2][X15=2][X16=0]
         [X17=0][X18=0][X19=1][X21=0,3][X22=0,1,3]
         [X24=0,1][X25=0][X26=0,2][X28=0][X29=4][X30=0]
         [X31=0][X32=0][X33=0][X34=0][X35=0]
         EVENTS COVERED  1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
3     2  [X1=0  4][X3=1  2][X4=0  1][X7=1,3][X8=1  2]                    31    9 8E+005
         [X9=0,1][X11=1][X13=0][X14=2][X15=2][X16=0]
         [X17=0][X18=0][X19=1][X21=1,2][X22=1,2][X23=0]
         [X24=0,1][X26=0][X27=0][X28=3][X29=4][X30=0]
         [X31=0][X32=0][X33=0][X34=0][X35=0,1]
         EVENTS COVERED  21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,
                         36,37,38,39,40,41,42,43,44,45,46,47
3   TOTALS                                                                    2 8E+006
    (2630 MS)

4     1  [X1=3  6][X2=0][X4=1  2][X8=1  2][X9=0,1]                        19    1 8E+006
         [X11=1][X12=1][X13=0][X14=2][X15=2][X16=0]
         [X17=0][X18=0][X19=1][X21=0,3][X22=0,1,3]
         [X24=0,1][X25=0][X26=0,2][X28=0][X29=4][X30=0]
         [X31=0][X32=0][X33=0][X34=0][X35=0]
         EVENTS COVERED  1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
4     2  [X1=0  4][X3=1  2][X4=0  1][X7=1,3][X8=1  2]                    36    9 8E+005
         [X9=0,1][X11=1][X13=0][X14=2][X15=2][X16=0]
         [X17=0][X18=0][X19=1][X21=1,2][X22=1,2][X23=0]
         [X24=0,1][X26=0][X27=0][X28=3][X29=4][X30=0]
         [X31=0][X32=0][X33=0][X34=0][X35=0,1]
         EVENTS COVERED  21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,
                         36,37,38,39,40,41,42,43,44,45,46,47
4   TOTALS                                                                    2 8E+006
    (3118 MS)

5     1  [X1=3  6][X2=0][X4=1  2][X8=1  2][X9=0,1]                        17    1 8E+006
```

the hierarchy are about to be generated by clustering with k varying from 2 to 4. The two seed events are arbitrarily selected as the first two events. The event numbers of the seeds are listed in the column marked SEEDS. The SPAR column gives the sparseness of each complex. The list event numbers of covered events is given following each class description.

This is the second iteration. Notice the decrease in total sparseness since the first iteration.

This is the third iterations. Another big improvement in total sparseness takes place.

The fourth iteration does not net an improvement in total sparseness. This iteration counts against the PROBE iterations count which is set by input declaration to 2.

Again no improvement in total

```
        [X11=1] [X12=1] [X13=0] [X14=2] [X15=2] [X16=0]
        [X17=0] [X18=0] [X19=1] [X21=0,3] [X22=0,1,3]
        [X24=0,1] [X25=0] [X26=0,2] [X28=0] [X29=4] [X30=0]
        [X31=0] [X32=0] [X33=0] [X34=0] [X35=0]
        EVENTS COVERED: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
 5   2  [X1=0  4] [X3=1  2] [X4=0  1] [X7=1,3] [X8=1  2]        46    9 8E+005
        [X9=0,1] [X11=1] [X13=0] [X14=2] [X15=2] [X16=0]
        [X17=0] [X18=0] [X19=1] [X21=1,2] [X22=1,2] [X23=0]
        [X24=0,1] [X26=0] [X27=0] [X28=3] [X29=4] [X30=0]
        [X31=0] [X32=0] [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED: 21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,
                        36,37,38,39,40,41,42,43,44,45,46,47
 5   TOTALS                                                          2 8E+006
    (2818 MS)

THE 2 BEST CLUSTERINGS FOLLOW ...   (13102 MS)

ITER/CLASS# VL-RULE                                                SEED --COSTS---
                                                                          SPAR
 3   1  [X1=3  6] [X2=0] [X4=1  2] [X8=1  2] [X9=0,1]               6    1 8E+006
        [X11=1] [X12=1] [X13=0] [X14=2] [X15=2] [X16=0]
        [X17=0] [X18=0] [X19=1] [X21=0,3] [X22=0,1,3]
        [X24=0,1] [X25=0] [X26=0,2] [X28=0] [X29=4] [X30=0]
        [X31=0] [X32=0] [X33=0] [X34=0] [X35=0]
        EVENTS COVERED: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
 3   2  [X1=0  4] [X3=1  2] [X4=0  1] [X7=1,3] [X8=1  2]        31    9 8E+005
        [X9=0,1] [X11=1] [X13=0] [X14=2] [X15=2] [X16=0]
        [X17=0] [X18=0] [X19=1] [X21=1,2] [X22=1,2] [X23=0]
        [X24=0,1] [X26=0] [X27=0] [X28=3] [X29=4] [X30=0]
        [X31=0] [X32=0] [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED: 21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,
                        36,37,38,39,40,41,42,43,44,45,46,47
 3   TOTALS                                                          2 8E+006

 2   1  [X3=2] [X4=0  1] [X7=0,1] [X8=1  2] [X9=0,1] [X11=1]       23    1 4E+006
        [X13=0] [X14=2] [X15=2] [X16=0] [X17=0] [X18=0]
        [X19=1] [X21=1,3] [X22=0,1] [X24=1] [X26=0] [X27=0]
        [X28=0,3] [X29=4] [X30=0] [X31=0] [X32=0] [X33=0]
        [X34=0] [X35=0,1]
        EVENTS COVERED: 1,2,3,4,5,6,7,8,9,10,21,22,23,24,25,26,27,28,29,30
 2   2  [X7=1,2,3] [X8=1  2] [X9=0,1] [X11=1] [X12=1]             31    1 4E+007
        [X13=0] [X14=2] [X15=2] [X16=0] [X17=0] [X18=0]
        [X19=1] [X21=0,1,2] [X22=2,3] [X23=0] [X24=0,1]
        [X25=0] [X26=0,2] [X28=0,3] [X29=4] [X30=0] [X31=0]
        [X32=0] [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED: 11,12,13,14,15,16,17,18,19,20,31,32,33,34,35,
                        36,37,38,39,40,41,42,43,44,45,46,47
 2   TOTALS                                                          1 5E+007

(15 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  2 2E+007

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1  K=3, CRITERION=F

INTERMEDIATE RESULTS...
ITER/CLASS# VL-RULE                                                SEED --COSTS---
                                                                          SPAR
 1   1  [X3=2] [X4=0  1] [X7=0,1] [X8=1  2] [X9=0,1] [X10=2]        1    1 1E+005
        [X11=1] [X13=0] [X14=2] [X15=2] [X16=0] [X17=0]
        [X18=0] [X19=1] [X21=1,3] [X22=1] [X24=1] [X26=0]
        [X27=0] [X28=0,3] [X29=4] [X30=0] [X31=0] [X32=0]
        [X33=0] [X34=0] [X35=0]
        EVENTS COVERED: 1,6,10,21,22,25,28,29
 1   2  [X3=2] [X4=0  1] [X7=0,1] [X8=1  2] [X9=0,1]               2    4 6E-005
        [X10=0  1] [X11=1] [X13=0] [X14=2] [X15=2] [X16=0]
```

sparseness. This is the second probe iteration and the program halts because the probe count is exhausted

The following summary report gives the two best clusterings found for k=2

This is the value for the formula S = sparseness $\times k^{**}\beta$, where β has the value 9

Now the process is repeated for k=9.

The first iteration

```
        [X17=0] [X18=0] [X19=1] [X21=1,3] [X22=1] [X24=1]
        [X26=0] [X27=0] [X28=0,3] [X29=4] [X30=0] [X31=0]
        [X32=0] [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED  2,5,7,23,24,26,27,30
1   3   [X8=1  2] [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2]      3     7 4E+007
        [X15=2] [X16=0] [X17=0] [X18=0] [X19=1] [X22=0,2,3]
        [X24=0,1] [X25=0] [X26=0,2] [X28=0,3] [X29=4]
        [X30=0] [X31=0] [X32=0] [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED  3,4,8,9,11,12,13,14,15,16,17,18,19,20,31,32,33,
                        34,35,36,37,38,39,40,41,42,43,44,45,46,47
1   TOTALS                                                        7 5E+007
    (3960 MS)

2   1   [X1=0  4] [X3=2] [X4=0] [X7=1] [X8=1  2] [X9=0,1]      22       2555   The second iteration The
        [X10=1  2] [X11=1] [X13=0] [X14=2] [X15=2] [X16=0]                     sparseness score improves
        [X17=0] [X18=0] [X19=1] [X21=1] [X22=1] [X23=0]
        [X24=1] [X25=1] [X26=0] [X27=0] [X28=3] [X29=4]
        [X30=0] [X31=0] [X32=0] [X33=0] [X34=0] [X35=0]
        EVENTS COVERED  22,24,26,28,29
2   2   [X3=2] [X4=0  1] [X7=0,1] [X8=1  2] [X9=0,1] [X11=1]   30     6 9E+005
        [X13=0] [X14=2] [X15=2] [X16=0] [X17=0] [X18=0]
        [X19=1] [X21=1,3] [X22=0,1] [X24=1] [X25=0] [X26=0]
        [X27=0] [X28=0,3] [X29=4] [X30=0] [X31=0] [X32=0]
        [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED  1,2,3,4,5,6,7,8,9,10,21,23,25,27,30
2   3   [X7=1,2,3] [X8=1  2] [X9=0,1] [X11=1] [X12=1]          31     1 4E+007
        [X13=0] [X14=2] [X15=2] [X16=0] [X17=0] [X18=0]
        [X19=1] [X21=0,1,2] [X22=2,3] [X23=0] [X24=0,1]
        [X25=0] [X26=0,2] [X28=0,3] [X29=4] [X30=0] [X31=0]
        [X32=0] [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED  11,12,13,14,15,16,17,18,19,20,31,32,33,34,35,
                        36,37,38,39,40,41,42,43,44,45,46,47
2   TOTALS                                                        1 5E+007
    (4767 MS)

3   1   [X1=0  4] [X3=2] [X4=0] [X7=1] [X8=1  2] [X9=0,1]      22     1 0E+004   The third iteration The
        [X10=1  2] [X11=1] [X13=0] [X14=2] [X15=2] [X16=0]                     sparseness score improves
        [X17=0] [X18=0] [X19=1] [X21=1] [X22=1] [X23=0]                        again.
        [X24=1] [X26=0] [X27=0] [X28=3] [X29=4] [X30=0]
        [X31=0] [X32=0] [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED  21,22,23,24,25,26,27,28,29,30
3   2   [X1=3  6] [X2=0] [X4=1  2] [X8=1  2] [X9=0,1]           2     1 8E+006
        [X11=1] [X12=1] [X13=0] [X14=2] [X15=2] [X16=0]
        [X17=0] [X18=0] [X19=1] [X21=0,3] [X22=0,1,3]
        [X24=0,1] [X25=0] [X26=0,2] [X28=0] [X29=4] [X30=0]
        [X31=0] [X32=0] [X33=0] [X34=0] [X35=0]
        EVENTS COVERED  1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
3   3   [X1=0  3] [X2=1] [X3=1  2] [X4=0  1] [X7=1,3]          31     2 5E+004
        [X8=1  2] [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2]
        [X15=2] [X16=0] [X17=0] [X18=0] [X19=1] [X21=1,2]
        [X22=2] [X23=0] [X24=0,1] [X25=0] [X26=0] [X27=0]
        [X28=3] [X29=4] [X30=0] [X31=0] [X32=0] [X33=0]
        [X34=0] [X35=1]
        EVENTS COVERED  31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47
3   TOTALS                                                        1 8E+006
    (4662 MS)

4   1   [X3=2] [X4=0  1] [X7=0,1] [X8=1  2] [X9=0,1] [X11=1]   22     6 9E+005   The fourth iteration The
        [X13=0] [X14=2] [X15=2] [X16=0] [X17=0] [X18=0]                        sparseness score again im-
        [X19=1] [X21=1,3] [X22=0,1] [X24=1] [X26=0] [X27=0]                    proves
        [X28=0,3] [X29=4] [X30=0] [X31=0] [X32=0] [X33=0]
        [X34=0] [X35=0]
        EVENTS COVERED  1,2,3,4,5,6,7,8,9,10,21,22,23,24,25,26,28,29,30
4   2   [X1=3  6] [X2=0] [X3=0] [X4=1  2] [X7=2,3] [X8=1]      19       1526
        [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2] [X15=2]
        [X16=0] [X17=0] [X18=0] [X19=1] [X21=0] [X22=3]
        [X23=0] [X24=0] [X25=0] [X26=2] [X27=1] [X28=0]
        [X29=4] [X30=0] [X31=0] [X32=0] [X33=0] [X34=0]
        [X35=0]
```

```
        EVENTS COVERED  11,12,13,14,15,16,17,18,19,20
4   3   [X1=0  3][X2=1][X3=1  2][X4=0  1][X7=1,3]          31     9.8E+004
        [X8=1  2][X9=0,1][X11=1][X13=0][X14=2][X15=2]
        [X16=0][X17=0][X18=0][X19=1][X21=1,2][X22=1,2]
        [X23=0][X24=0,1][X25=0][X26=0][X27=0][X28=3]
        [X29=4][X30=0][X31=0][X32=0][X33=0][X34=0]
        [X35=1]
        EVENTS COVERED  27,31,32,33,34,35,36,37,38,39,40,41,42,43,44,
                        45,46,47
4   TOTALS                                                  7  9E+005
    (5362 MS)

5   1   [X1=3  6][X2=0][X3=2][X4=0  1][X7=0,1][X8=1  2]     6     2.0E+005   The fifth iteration. The
        [X9=0,1][X11=1][X13=0][X14=2][X15=2][X16=0]                          sparseness score improves a
        [X17=0][X18=0][X19=1][X21=1,3][X22=0,1][X24=1]                       little
        [X26=0][X27=0][X28=0,3][X29=4][X30=0][X31=0]
        [X32=0][X33=0][X34=0][X35=0]
        EVENTS COVERED  1,2,3,4,5,6,7,8,9,10,26,29
5   2   [X1=3  6][X2=0][X3=0][X4=1  2][X7=2,3][X8=1]        19          1526
        [X9=0,1][X11=1][X12=1][X13=0][X14=2][X15=2]
        [X16=0][X17=0][X18=0][X19=1][X21=0][X22=3]
        [X23=0][X24=0][X25=0][X26=2][X27=1][X28=0]
        [X29=4][X30=0][X31=0][X32=0][X33=0][X34=0]
        [X35=0]
        EVENTS COVERED  11,12,13,14,15,16,17,18,19,20
5   3   [X1=0  3][X2=1][X3=1..2][X4=0..1][X7=1,3]          31     3.9E+005
        [X8=1..2][X9=0,1][X11=1][X13=0][X14=2][X15=2]
        [X16=0][X17=0][X18=0][X19=1][X21=1,2][X22=1,2]
        [X23=0][X24=0,1][X26=0][X27=0][X28=3][X29=4]
        [X30=0][X31=0][X32=0][X33=0][X34=0][X35=0,1]
        EVENTS COVERED  21,22,23,24,25,27,28,30,31,32,33,34,35,36,37,
                        38,39,40,41,42,43,44,45,46,47
5   TOTALS                                                  5  9E+005
    (5637 MS)

6   1   [X1=3..6][X2=0][X3=2][X4=0..1][X7=0,1][X8=1..2]     6     2.0E+005   The sixth iteration. No im-
        [X9=0,1][X11=1][X13=0][X14=2][X15=2][X16=0]                          provement in sparseness.
        [X17=0][X18=0][X19=1][X21=1,3][X22=0,1][X24=1]                       This iteration counts against
        [X26=0][X27=0][X28=0,3][X29=4][X30=0][X31=0]                         the PROBE count
        [X32=0][X33=0][X34=0][X35=0]
        EVENTS COVERED  1,2,3,4,5,6,7,8,9,10,26,29
6   2   [X1=3..6][X2=0][X3=0][X4=1..2][X7=2,3][X8=1]        19          1526
        [X9=0,1][X11=1][X12=1][X13=0][X14=2][X15=2]
        [X16=0][X17=0][X18=0][X19=1][X21=0][X22=3]
        [X23=0][X24=0][X25=0][X26=2][X27=1][X28=0]
        [X29=4][X30=0][X31=0][X32=0][X33=0][X34=0]
        [X35=0]
        EVENTS COVERED  11,12,13,14,15,16,17,18,19,20
6   3   [X1=0..3][X2=1][X3=1..2][X4=0..1][X7=1,3]          36     3.9E+005
        [X8=1..2][X9=0,1][X11=1][X13=0][X14=2][X15=2]
        [X16=0][X17=0][X18=0][X19=1][X21=1,2][X22=1,2]
        [X23=0][X24=0,1][X26=0][X27=0][X28=3][X29=4]
        [X30=0][X31=0][X32=0][X33=0][X34=0][X35=0,1]
        EVENTS COVERED  21,22,23,24,25,27,28,30,31,32,33,34,35,36,37,
                        38,39,40,41,42,43,44,45,46,47
6   TOTALS                                                  5  9E+005
    (4514 MS)

7   1   [X1=3..6][X2=0][X3=2][X4=0..1][X7=0,1][X8=1..2]     7     2.0E+005   The seventh iteration. The
        [X9=0,1][X11=1][X13=0][X14=2][X15=2][X16=0]                          sparseness does not improve.
        [X17=0][X18=0][X19=1][X21=1,3][X22=0,1][X24=1]                       This exhausts the PROBE
        [X26=0][X27=0][X28=0,3][X29=4][X30=0][X31=0]                         count.
        [X32=0][X33=0][X34=0][X35=0]
        EVENTS COVERED  1,2,3,4,5,6,7,8,9,10,26,29
7   2   [X1=3  6][X2=0][X3=0][X4=1  2][X7=2,3][X8=1]        18          1526
        [X9=0,1][X11=1][X12=1][X13=0][X14=2][X15=2]
        [X16=0][X17=0][X18=0][X19=1][X21=0][X22=3]
        [X23=0][X24=0][X25=0][X26=2][X27=1][X28=0]
        [X29=4][X30=0][X31=0][X32=0][X33=0][X34=0]
```

```
        [X35=0]
        EVENTS COVERED: 11,12,13,14,15,16,17,18,19,20
7   3   [X1=0  3] [X2=1] [X3=1  2] [X4=0  1] [X7=1,3]          46    3 9E+005
        [X8=1  2] [X9=0,1] [X11=1] [X13=0] [X14=2] [X15=2]
        [X16=0] [X17=0] [X18=0] [X19=1] [X21=1,2] [X22=1,2]
        [X23=0] [X24=0,1] [X26=0] [X27=0] [X28=3] [X29=4]
        [X30=0] [X31=0] [X32=0] [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED: 21,22,23,24,25,27,28,30,31,32,33,34,35,36,37,
                        38,39,40,41,42,43,44,45,46,47
7   TOTALS                                                          5 9E+005
  (4826 MS)


THE 2 BEST CLUSTERINGS FOLLOW      (34746 MS)

ITER CLASS# VL-RULE                                          SEED --COSTS---
                                                                      SPAR
5    1   [X1=3  6] [X2=0] [X3=2] [X4=0  1] [X7=0,1] [X8=1  2]    6    2 0E+005
        [X9=0,1] [X11=1] [X13=0] [X14=2] [X15=2] [X16=0]
        [X17=0] [X18=0] [X19=1] [X21=1,3] [X22=0,1] [X24=1]
        [X26=0] [X27=0] [X28=0,3] [X29=4] [X30=0] [X31=0]
        [X32=0] [X33=0] [X34=0] [X35=0]
        EVENTS COVERED: 1,2,3,4,5,6,7,8,9,10,26,29
5    2   [X1=3  6] [X2=0] [X3=0] [X4=1  2] [X7=2,3] [X8=1]      19        1526
        [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2] [X15=2]
        [X16=0] [X17=0] [X18=0] [X19=1] [X21=0] [X22=3]
        [X23=0] [X24=0] [X25=0] [X26=2] [X27=1] [X28=0]
        [X29=4] [X30=0] [X31=0] [X32=0] [X33=0] [X34=0]
        [X35=0]
        EVENTS COVERED: 11,12,13,14,15,16,17,18,19,20
5    3   [X1=0  3] [X2=1] [X3=1  2] [X4=0  1] [X7=1,3]          31    3 9E+005
        [X8=1  2] [X9=0,1] [X11=1] [X13=0] [X14=2] [X15=2]
        [X16=0] [X17=0] [X18=0] [X19=1] [X21=1,2] [X22=1,2]
        [X23=0] [X24=0,1] [X26=0] [X27=0] [X28=3] [X29=4]
        [X30=0] [X31=0] [X32=0] [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED: 21,22,23,24,25,27,28,30,31,32,33,34,35,36,37,
                        38,39,40,41,42,43,44,45,46,47
5    TOTALS                                                          5 9E+005

4    1   [X3=2] [X4=0  1] [X7=0,1] [X8=1  2] [X9=0,1] [X11=1]    2.    6 9E+005
        [X13=0] [X14=2] [X15=2] [X16=0] [X17=0] [X18=0]
        [X19=1] [X21=1,3] [X22=0,1] [X24=1] [X26=0] [X27=0]
        [X28=0,3] [X29=4] [X30=0] [X31=0] [X32=0] [X33=0]
        [X34=0] [X35=0]
        EVENTS COVERED: 1,2,3,4,5,6,7,8,9,10,21,22,23,24,25,26,28,29,30
4    2   [X1=3  6] [X2=0] [X3=0] [X4=1  2] [X7=2,3] [X8=1]      19        1526
        [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2] [X15=2]
        [X16=0] [X17=0] [X18=0] [X19=1] [X21=0] [X22=3]
        [X23=0] [X24=0] [X25=0] [X26=2] [X27=1] [X28=0]
        [X29=4] [X30=0] [X31=0] [X32=0] [X33=0] [X34=0]
        [X35=0]
        EVENTS COVERED: 11,12,13,14,15,16,17,18,19,20
4    3   [X1=0  3] [X2=1] [X3=1  2] [X4=0  1] [X7=1,3]          31    9 8E+004
        [X8=1  2] [X9=0,1] [X11=1] [X13=0] [X14=2] [X15=2]
        [X16=0] [X17=0] [X18=0] [X19=1] [X21=1,2] [X22=1,2]
        [X23=0] [X24=0,1] [X25=0] [X26=0] [X27=0] [X28=3]
        [X29=4] [X30=0] [X31=0] [X32=0] [X33=0] [X34=0]
        [X35=1]
        EVENTS COVERED: 27,31,32,33,34,35,36,37,38,39,40,41,42,43,44,
                        45,46,47
4    TOTALS                                                          7 9E+005

(42 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  1.6E+007

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

The two best clusterings with k=9 are reported

*The value of the expression sparseness × k**3*

```
EXPERIMENT 1  K=4, CRITERION=F                                          The whole process is repeated
                                                                        for k=4
INTERMEDIATE RESULTS
ITER/ CLASS# VL-RULE                                          SEED --COSTS---
                                                                        SPAR
1     1   [X3=2][X4=0  1][X7=0,1][X8=1  2][X9=0][X11=1]        1     3 4E+005   The first iteration
          [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]
          [X19=1][X21=1,3][X22=1][X24=1][X26=0][X27=0]
          [X28=0,3][X29=4][X30=0][X31=0][X32=0][X33=0]
          [X34=0][X35=0,1]
          EVENTS COVERED: 1,6,7,10,22,23,26,27,28,29
1     2   [X1=4  6][X2=0][X4=1  2][X6=1  3][X7=0,1,3]          2     1 9E+005
          [X8=1][X9=1][X11=1][X12=1][X13=0][X14=2][X15=2]
          [X16=0][X17=0][X18=0][X19=1][X21=0,3][X22=0,1,3]
          [X24=0,1][X25=0][X26=0,2][X28=0][X29=4]
          [X30=0][X31=0][X32=0][X33=0][X34=0][X35=0]
          EVENTS COVERED: 2,5,9,11,12,15,17,19
1     3   [X1=0  3][X3=1  2][X4=0  1][X7=0,1,3][X8=1  2]       3     5 3E+006
          [X9=1][X11=1][X13=0][X14=2][X15=2][X16=0]
          [X17=0][X18=0][X19=1][X21=1,2,3][X22=0,1,2]
          [X24=0,1][X26=0][X27=0][X28=0,3][X29=4][X30=0]
          [X31=0][X32=0][X33=0][X34=0][X35=0,1]
          EVENTS COVERED: 3,8,21,24,25,30,31,34,35,37,38,41,43,44,47
1     4   [X7=1,2][X8=1  2][X9=0][X11=1][X12=1][X13=0]         4     1 4E+007
          [X14=2][X15=2][X16=0][X17=0][X18=0][X19=1]
          [X21=0,1,3][X22=0,2,3][X24=0,1][X25=0][X26=0,2]
          [X28=0,3][X29=4][X30=0][X31=0][X32=0][X33=0]
          [X34=0][X35=0,1]
          EVENTS COVERED: 4,13,14,16,18,20,32,33,36,39,40,42,45,46
1           EXCEPTIONAL EVENTS: 22,27                                 These events could not be
1   TOTALS                                                    2 0E+007  covered and still maintain dis-
    (7267 MS)                                                           joint complexes

2     1   [X1=0  4][X3=2][X4=0  1][X7=1][X8=1  2][X9=0,1]     23     1 2E+005   The second iteration The
          [X11=1][X13=0][X14=2][X15=2][X16=0][X17=0]                    sparseness score improves
          [X18=0][X19=1][X21=1][X22=1,2][X23=0][X24=0,1]
          [X26=0][X27=0][X28=3][X29=4][X30=0][X31=0]
          [X32=0][X33=0][X34=0][X35=0,1]
          EVENTS COVERED: 21,22,23,24,25,26,27,28,29,30,32,33,36,40,45,46
2     2   [X1=3  6][X2=0][X3=0][X4=1  2][X7=2,3][X8=1]         19          1526
          [X9=0,1][X11=1][X12=1][X13=0][X14=2][X15=2]
          [X16=0][X17=0][X18=0][X19=1][X21=0][X22=3]
          [X23=0][X24=0][X25=0][X26=2][X27=1][X28=0]
          [X29=4][X30=0][X31=0][X32=0][X33=0][X34=0]
          [X35=0]
          EVENTS COVERED: 11,12,13,14,15,16,17,18,19,20
2     3   [X1=1  6][X3=2][X4=0  1][X7=0,1,3][X8=1  2]         31     1 3E+006
          [X9=0,1][X11=1][X12=1][X13=0][X14=2][X15=2]
          [X16=0][X17=0][X18=0][X19=1][X21=2,3][X22=0,1,
          2][X24=0,1][X25=0][X26=0][X27=0][X28=0,3]
          [X29=4][X30=0][X31=0][X32=0][X33=0][X34=0]
          [X35=0,1]
          EVENTS COVERED: 1,2,3,4,5,6,7,8,9,10,31,34,35,37,38,43,44
2     4   [X1=0  3][X2=1][X3=1][X4=0  1][X6=1  3][X7=1]       39          4604
          [X8=1  2][X9=0,1][X11=1][X12=1][X13=0][X14=2]
          [X15=2][X16=0][X17=0][X18=0][X19=1][X21=1,2]
          [X22=2][X23=0][X24=0,1][X25=0][X26=0][X27=0]
          [X28=3][X29=4][X30=0][X31=0][X32=0][X33=0]
          [X34=0][X35=1]
          EVENTS COVERED: 39,41,42,47
2   TOTALS                                                    1 5E+006
    (10748 MS)

3     1   [X1=0  4][X3=2][X4=0][X7=1][X8=1  2][X9=0,1]       23     1 0E+004   The third iteration The
          [X10=1  2][X11=1][X13=0][X14=2][X15=2][X16=0]                 sparseness score improves
          [X17=0][X18=0][X19=1][X21=1][X22=1][X23=0]                    again
          [X24=1][X26=0][X27=0][X28=3][X29=4][X30=0]
          [X31=0][X32=0][X33=0][X34=0][X35=0,1]
          EVENTS COVERED: 21,22,23,24,25,26,27,28,29,30
```

```
3    2  [X1=3  6] [X2=0] [X3=0] [X4=1  2] [X7=2,3] [X8=1]          19          1526
        [X9=0,1] [X11=1] [X12=1,] [X13=0] [X14=2] [X15=2]
        [X16=0] [X17=0] [X18=0] [X19=1] [X21=0] [X22=3]
        [X23=0] [X24=0] [X25=0] [X26=2] [X27=1] [X28=0]
        [X29=4] [X30=0] [X31=0] [X32=0] [X33=0] [X34=0]
        [X35=0]
        EVENTS COVERED: 11,12,13,14,15,16,17,18,19,20
3    3  [X1=3  6] [X2=0] [X3=2] [X4=1] [X6=1  3] [X7=0,1]          8           2294
        [X8=1  2] [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2]
        [X15=2] [X16=0] [X17=0] [X18=0] [X19=1] [X21=3]
        [X22=0,1] [X23=1] [X24=1] [X25=0] [X26=0] [X27=0]
        [X28=0] [X29=4] [X30=0] [X31=0] [X32=0] [X33=0]
        [X34=0] [X35=0]
        EVENTS COVERED: 1,2,3,4,5,6,7,8,9,10
3    4  [X1=0  3] [X2=1] [X3=1  2] [X4=0  1] [X7=1,3]           47          2 5E+004
        [X8=1  2] [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2]
        [X15=2] [X16=0] [X17=0] [X18=0] [X19=1] [X21=1,2]
        [X22=2] [X23=0] [X24=0,1] [X25=0] [X26=0] [X27=0]
        [X28=3] [X29=4] [X30=0] [X31=0] [X32=0] [X33=0]
        [X34=0] [X35=1]
        EVENTS COVERED: 31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47
3    TOTALS                                                          3 9E+004
     (8878 MS)

4    1  [X1=0  4] [X3=2] [X4=0] [X7=1] [X8=1  2] [X9=0,1]          22          1 0E+004   The fourth iteration. The
        [X10=1  2] [X11=1] [X13=0] [X14=2] [X15=2] [X16=0]                                 sparseness does not improve
        [X17=0] [X18=0] [X19=1] [X21=1] [X22=1] [X23=0]
        [X24=1] [X26=0] [X27=0] [X28=3] [X29=4] [X30=0]
        [X31=0] [X32=0] [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED: 21,22,23,24,25,26,27,28,29,30
4    2  [X1=3  6] [X2=0] [X3=0] [X4=1  2] [X7=2,3] [X8=1]          19          1526
        [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2] [X15=2]
        [X16=0] [X17=0] [X18=0] [X19=1] [X21=0] [X22=3]
        [X23=0] [X24=0] [X25=0] [X26=2] [X27=1] [X28=0]
        [X29=4] [X30=0] [X31=0] [X32=0] [X33=0] [X34=0]
        [X35=0]
        EVENTS COVERED: 11,12,13,14,15,16,17,18,19,20
4    3  [X1=3  6] [X2=0] [X3=2] [X4=1] [X6=1  3] [X7=0,1]          2           2294
        [X8=1  2] [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2]
        [X15=2] [X16=0] [X17=0] [X18=0] [X19=1] [X21=3]
        [X22=0,1] [X23=1] [X24=1] [X25=0] [X26=0] [X27=0]
        [X28=0] [X29=4] [X30=0] [X31=0] [X32=0] [X33=0]
        [X34=0] [X35=0]
        EVENTS COVERED: 1,2,3,4,5,6,7,8,9,10
4    4  [X1=0  3] [X2=1] [X3=1  2] [X4=0  1] [X7=1,3]           31          2 5E+004
        [X8=1  2] [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2]
        [X15=2] [X16=0] [X17=0] [X18=0] [X19=1] [X21=1,2]
        [X22=2] [X23=0] [X24=0,1] [X25=0] [X26=0] [X27=0]
        [X28=3] [X29=4] [X30=0] [X31=0] [X32=0] [X33=0]
        [X34=0] [X35=1]
        EVENTS COVERED: 31,32,33,34 35,36,37,38,39,40,41,42,43,44,45,46,47
4    TOTALS                                                          3 9E+004
     (8041 MS)

5    1  [X1=0  4] [X3=2] [X4=0] [X7=1] [X8=1  2] [X9=0,1]          21          1 0E+004   The fifth iteration. The
        [X10=1  2] [X11=1] [X13=0] [X14=2] [X15=2] [X16=0]                                 sparseness does not improve.
        [X17=0] [X18=0] [X19=1] [X21=1] [X22=1] [X23=0]                                    This exhausts the PROBE
        [X24=1] [X26=0] [X27=0] [X28=3] [X29=4] [X30=0]                                    count.
        [X31=0] [X32=0] [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED: 21,22,23,24,25,26,27,28,29,30
5    2  [X1=3  6] [X2=0] [X3=0] [X4=1  2] [X7=2,3] [X8=1]          18          1526
        [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2] [X15=2]
        [X16=0] [X17=0] [X18=0] [X19=1] [X21=0] [X22=3]
        [X23=0] [X24=0] [X25=0] [X26=2] [X27=1] [X28=0]
        [X29=4] [X30=0] [X31=0] [X32=0] [X33=0] [X34=0]
        [X35=0]
        EVENTS COVERED: 11,12,13,14,15,16,17,18,19,20
5    3  [X1=3  6] [X2=0] [X3=2] [X4=1] [X6=1  3] [X7=0,1]          10          2294
        [X8=1  2] [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2]
```

```
        [X15=2] [X16=0] [X17=0] [X18=0] [X19=1] [X21=3]
        [X22=0,1] [X23=1] [X24=1] [X25=0] [X26=0] [X27=0]
        [X28=0] [X29=4] [X30=0] [X31=0] [X32=0] [X33=0]
        [X34=0] [X35=0]
        EVENTS COVERED: 1,2,3,4,5,6,7,8,9,10
5   4   [X1=0  3] [X2=1] [X3=1  2] [X4=0  1] [X7=1,3]          46    2.5E+004
        [X8=1  2] [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2]
        [X15=2] [X16=0] [X17=0] [X18=0] [X19=1] [X21=1,2]
        [X22=2] [X23=0] [X24=0,1] [X25=0] [X26=0] [X27=0]
        [X28=3] [X29=4] [X30=0] [X31=0] [X32=0] [X33=0]
        [X34=0] [X35=1]
        EVENTS COVERED: 31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47
5   TOTALS                                                        3.9E+004
    (9246 MS)


THE 2 BEST CLUSTERINGS FOLLOW...   (44984 MS)

ITER/CLASS# VL-RULE                                          SEED --COSTS---
                                                                      SPAR
3   1   [X1=0  4] [X3=2] [X4=0] [X7=1] [X8=1  2] [X9=0,1]     23    1.0E+004
        [X10=1  2] [X11=1] [X13=0] [X14=2] [X15=2] [X16=0]
        [X17=0] [X18=0] [X19=1] [X21=1] [X22=1] [X23=0]
        [X24=1] [X26=0] [X27=0] [X28=3] [X29=4] [X30=0]
        [X31=0] [X32=0] [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED: 21,22,23,24,25,26,27,28,29,30
3   2   [X1=3  6] [X2=0] [X3=0] [X4=1  2] [X7=2,3] [X8=1]      19        1526
        [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2] [X15=2]
        [X16=0] [X17=0] [X18=0] [X19=1] [X21=0] [X22=3]
        [X23=0] [X24=0] [X25=0] [X26=2] [X27=1] [X28=0]
        [X29=4] [X30=0] [X31=0] [X32=0] [X33=0] [X34=0]
        [X35=0]
        EVENTS COVERED: 11,12,13,14,15,16,17,18,19,20
3   3   [X1=3  6] [X2=0] [X3=2] [X4=1] [X6=1  3] [X7=0,1]      8         2294
        [X8=1  2] [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2]
        [X15=2] [X16=0] [X17=0] [X18=0] [X19=1] [X21=3]
        [X22=0,1] [X23=1] [X24=1] [X25=0] [X26=0] [X27=0]
        [X28=0] [X29=4] [X30=0] [X31=0] [X32=0] [X33=0]
        [X34=0] [X35=0]
        EVENTS COVERED: 1,2,3,4,5,6,7,8,9,10
3   4   [X1=0  3] [X2=1] [X3=1  2] [X4=0  1] [X7=1,3]          47    2.5E+004
        [X8=1  2] [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2]
        [X15=2] [X16=0] [X17=0] [X18=0] [X19=1] [X21=1,2]
        [X22=2] [X23=0] [X24=0,1] [X25=0] [X26=0] [X27=0]
        [X28=3] [X29=4] [X30=0] [X31=0] [X32=0] [X33=0]
        [X34=0] [X35=1]
        EVENTS COVERED: 31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47
3   TOTALS                                                        3.9E+004

2   1   [X1=0  4] [X3=2] [X4=0  1] [X7=1] [X8=1  2] [X9=0,1]    23    1.2E+005
        [X11=1] [X13=0] [X14=2] [X15=2] [X16=0] [X17=0]
        [X18=0] [X19=1] [X21=1] [X22=1,2] [X23=0] [X24=0,1]
        [X26=0] [X27=0] [X28=3] [X29=4] [X30=0] [X31=0]
        [X32=0] [X33=0] [X34=0] [X35=0,1]
        EVENTS COVERED: 21,22,23,24,25,26,27,28,29,30,32,33,36,40,45,46
2   2   [X1=3  6] [X2=0] [X3=0] [X4=1  2] [X7=2,3] [X8=1]      19        1526
        [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2] [X15=2]
        [X16=0] [X17=0] [X18=0] [X19=1] [X21=0] [X22=3]
        [X23=0] [X24=0] [X25=0] [X26=2] [X27=1] [X28=0]
        [X29=4] [X30=0] [X31=0] [X32=0] [X33=0] [X34=0]
        [X35=0]
        EVENTS COVERED: 11,12,13,14,15,16,17,18,19,20
2   3   [X1=1  6] [X3=2] [X4=0  1] [X7=0,1,3] [X8=1  2]       31    1.3E+006
        [X9=0,1] [X11=1] [X12=1] [X13=0] [X14=2] [X15=2]
        [X16=0] [X17=0] [X18=0] [X19=1] [X21=2,3] [X22=0,1,
        2] [X24=0,1] [X25=0] [X26=0] [X27=0] [X28=0,3]
        [X29=4] [X30=0] [X31=0] [X32=0] [X33=0] [X34=0]
        [X35=0,1]
```

The two best clusterings for k=4 are reported. This solution matches human classifications of diseased soybeans. It forms the top level of the classification hierarchy since it is the best solution as measured by heuristic S.

```
        EVENTS COVERED: 1,2,3,4,5,6,7,8,9,10,31,34,35,37,38,43,44
 2    4 [X1=0..3][X2=1][X3=1][X4=0..1][X6=1..3][X7=1]      39        4604
        [X8=1..2][X9=0,1][X11=1][X12=1][X13=0][X14=2]
        [X15=2][X16=0][X17=0][X18=0][X19=1][X21=1,2]
        [X22=2][X23=0][X24=0,1][X25=0][X26=0][X27=0]
        [X28=3][X29=4][X30=0][X31=0][X32=0][X33=0]
        [X34=0][X35=1]
        EVENTS COVERED: 39,41,42,47
 2   TOTALS                                                 1.5E+006

(50 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  2.5E+006

WITH BETA=  3.00 THE BEST CLUSTERING AT THIS LEVEL IS FOR K=4

BEGINNING CLASSIFICATION BELOW HIERARCHY PATH 1-0

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1: K=2, CRITERION=F

THE 1 BEST CLUSTERINGS FOLLOW...   (2242 MS)

ITER/CLASS# VL-RULE                                        SEED --COSTS---
                                                                    SPAR
 1    1 [X1=0][X2=1][X3=2][X4=0][X5=0][X6=0..2][X7=1]      21          91
        [X8=1..2][X9=0,1][X10=1..2][X11=1][X12=0]
        [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]
        [X19=1][X20=0][X21=1][X22=1][X23=0][X24=1]
        [X26=0][X27=0][X28=3][X29=4][X30=0][X31=0]
        [X32=0][X33=0][X34=0][X35=0,1]
        EVENTS COVERED: 21,24,25,27,30
 1    2 [X1=2..4][X3=2][X4=0][X7=1][X8=1..2][X9=0]         22        1531
        [X10=1..2][X11=1][X13=0][X14=2][X15=2][X16=0]
        [X17=0][X18=0][X19=1][X21=1][X22=1][X23=0]
        [X24=1][X26=0][X27=0][X28=3][X29=4][X30=0]
        [X31=0][X32=0][X33=0][X34=0][X35=0]
        EVENTS COVERED: 22,23,26,28,29
 1   TOTALS                                                 1622

(12 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  1.3E+004

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1: K=3, CRITERION=F

THE 1 BEST CLUSTERINGS FOLLOW...   (4697 MS)

ITER/CLASS# VL-RULE                                        SEED --COSTS---
                                                                    SPAR
 2    1 [X1=0][X2=1][X3=2][X4=0][X5=0][X6=0..2][X7=1]      21          91
        [X8=1..2][X9=0,1][X10=1..2][X11=1][X12=0]
        [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]
        [X19=1][X20=0][X21=1][X22=1][X23=0][X24=1]
        [X26=0][X27=0][X28=3][X29=4][X30=0][X31=0]
        [X32=0][X33=0][X34=0][X35=0,1]
        EVENTS COVERED: 21,24,25,27,30
 2    2 [X1=2][X2=1][X3=2][X4=0][X5=0][X6=2..3][X7=1]      22          13
        [X8=1..2][X9=0][X10=1..2][X11=1][X12=0][X13=0]
        [X14=2][X15=2][X16=0][X17=0][X18=0][X19=1]
        [X20=0][X21=1][X22=1][X23=0][X24=1][X26=0]
```

The value of the expression sparseness × $k^{**}\beta$ is given

The lowest value of S denoting the best fit was obtained with k=4.

The top level of the hierarchy has four classes. The cluster algorithm is applied to each class individually to generate second-level classes

For class 1, clusterings are generated for k varying from 2 to 4. The detailed output of intermediate and alternative results has been suppressed in this part of the listing to conserve space.

This solution is best according to heuristic S for the second level classes which split class 1.

```
          [X27=0] [X28=3] [X29=4] [X30=0] [X31=0] [X32=0]
          [X33=0] [X34=0] [X35=0]
          EVENTS COVERED: 22,23,28
 2    3   [X1=3  4] [X2=0] [X3=2] [X4=0] [X5=1] [X7=1] [X8=2]      28          30
          [X9=0] [X10=1  2] [X11=1] [X13=0] [X14=2] [X15=2]
          [X16=0] [X17=0] [X18=0] [X19=1] [X20=1] [X21=1]
          [X22=1] [X23=0] [X24=1] [X25=1] [X26=0] [X27=0]
          [X28=3] [X29=4] [X30=0] [X31=0] [X32=0] [X33=0]
          [X34=0] [X35=0]
          EVENTS COVERED: 26,29
 2   TOTALS                                                        134

(24 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  3 6E+003

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1  K=4, CRITERION=F

THE 1 BEST CLUSTERINGS FOLLOW.    (12285 MS)

ITER CLASS# VL-RULE                                              SEED --COSTS---
                                                                       SPAR
 5    1   [X1=0] [X2=1] [X3=2] [X4=0] [X5=0] [X6=0  2] [X7=1]      30          20
          [X8=1  2] [X9=1] [X10=1  2] [X11=1] [X12=0] [X13=0]
          [X14=2] [X15=2] [X16=0] [X17=0] [X18=0] [X19=1]
          [X20=0] [X21=1] [X22=1] [X23=0] [X24=1] [X26=0]
          [X27=0] [X28=3] [X29=4] [X30=0] [X31=0] [X32=0]
          [X33=0] [X34=0] [X35=0]
          EVENTS COVERED: 21,24,25,30
 5    2   [X1=2  4] [X3=2] [X4=0] [X6=0  2] [X7=1] [X8=1  2]      22          286
          [X9=0] [X10=2] [X11=1] [X13=0] [X14=2] [X15=2]
          [X16=0] [X17=0] [X18=0] [X19=1] [X21=1] [X22=1]
          [X23=0] [X24=1] [X25=1] [X26=0] [X27=0] [X28=3]
          [X29=4] [X30=0] [X31=0] [X32=0] [X33=0] [X34=0]
          [X35=0]
          EVENTS COVERED: 22,29
 5    3   [X1=2  3] [X3=2] [X4=0] [X6=3] [X7=1] [X8=2] [X9=0]      23          61
          [X10=1  2] [X11=1] [X12=0] [X13=0] [X14=2] [X15=2]
          [X16=0] [X17=0] [X18=0] [X19=1] [X21=1] [X22=1]
          [X23=0] [X24=1] [X26=0] [X27=0] [X28=3] [X29=4]
          [X30=0] [X31=0] [X32=0] [X33=0] [X34=0] [X35=0]
          EVENTS COVERED: 23,26,28
 5    4   [X1=0] [X2=1] [X3=2] [X4=0] [X5=0] [X6=0] [X7=1]        27           0
          [X8=1] [X9=0] [X10=1] [X11=1] [X12=0] [X13=0] [X14=2]
          [X15=2] [X16=0] [X17=0] [X18=0] [X19=1] [X20=0]
          [X21=1] [X22=1] [X23=0] [X24=1] [X25=0] [X26=0]
          [X27=0] [X28=3] [X29=4] [X30=0] [X31=0] [X32=0]
          [X33=0] [X34=0] [X35=1]
          EVENTS COVERED: 27
 5   TOTALS                                                        367

(70 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  2 3E-004

WITH BETA=  3 00 THE BEST CLUSTERING AT THIS LEVEL IS FOR K=3

BEGINNING CLASSIFICATION BELOW HIERARCHY PATH 2-0

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1.  K=2, CRITERION=F

THE 1 BEST CLUSTERINGS FOLLOW      (2036 MS)

ITER CLASS# VL-RULE                                              SEED --COSTS---
                                                                       SPAR
```

The first class is best subdivided into 3 subclasses.

```
 2    1  [X1=4..6][X2=0][X3=0][X4=1..2][X6=1..3][X7=3]      11        211
         [X8=1][X9=1][X11=1][X12=1][X13=0][X14=2][X15=2]
         [X16=0][X17=0][X18=0][X19=1][X21=0][X22=3]
         [X23=0][X24=0][X25=0][X26=2][X27=1][X28=0]
         [X29=4][X30=0][X31=0][X32=0][X33=0][X34=0]
         [X35=0]
         EVENTS COVERED: 11,12,15,17,19
 2    2  [X1=3..6][X2=0][X3=0][X4=1..2][X7=2][X...=1]       16        379
         [X9=0][X11=1][X12=1][X13=0][X14=2][X15=2]
         [X16=0][X17=0][X18=0][X19=1][X21=0][X22=3]
         [X23=0][X24=0][X25=0][X26=2][X27=1][X28=0]
         [X29=4][X30=0][X31=0][X32=0][X33=0][X34=0]
         [X35=0]
         EVENTS COVERED: 13,14,16,18,20
 2   TOTALS                                                   590

(12 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  4.7E+003

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1: K=3, CRITERION=F

THE 1 BEST CLUSTERINGS FOLLOW...   (5984 MS)

ITER/CLASS# VL-RULE                                          SEED --COSTS---
                                                                      SPAR
 4    1  [X1=3..5][X2=0][X3=0][X4=2][X7=2][X8=1][X9=0]      16         93
         [X10=1..2][X11=1][X12=1][X13=0][X14=2][X15=2]
         [X16=0][X17=0][X18=0][X19=1][X21=0][X22=3]
         [X23=0][X24=0][X25=0][X26=2][X27=1][X28=0]
         [X29=4][X30=0][X31=0][X32=0][X33=0][X34=0]
         [X35=0]
         EVENTS COVERED: 13,16,18
 4    2  [X1=3..6][X2=0][X3=0][X4=1..2][X5=0][X6=1]         14         30
         [X7=2,3][X8=1][X9=0,1][X10=0][X11=1][X12=1]
         [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]
         [X19=1][X20=0][X21=0][X22=3][X23=0][X24=0]
         [X25=0][X26=2][X27=1][X28=0][X29=4][X30=0]
         [X31=0][X32=0][X33=0][X34=0][X35=0]
         EVENTS COVERED: 12,14
 4    3  [X1=4..6][X2=0][X3=0][X4=1..2][X5=1][X6=1..3]      17        105
         [X7=3][X8=1][X9=1][X11=1][X12=1][X13=0][X14=2]
         [X15=2][X16=0][X17=0][X18=0][X19=1][X21=0]
         [X22=3][X23=0][X24=0][X25=0][X26=2][X27=1]
         [X28=0][X29=4][X30=0][X31=0][X32=0][X33=0]
         [X34=0][X35=0]
         EVENTS COVERED: 11,15,17
 4       EXCEPTIONAL EVENTS: 19,20
 4   TOTALS                                                   228

(36 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  6.2E+003

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1: K=4, CRITERION=F

THE 1 BEST CLUSTERINGS FOLLOW...   (15354 MS)

ITER/CLASS# VL-RULE                                          SEED --COSTS---
                                                                      SPAR
 2    1  [X1=5][X2=0][X3=0][X4=2][X6=3][X7=2,3][X8=1]       11          6
         [X9=0,1][X10=2][X11=1][X12=1][X13=0][X14=2]
         [X15=2][X16=0][X17=0][X18=0][X19=1][X20=0]
         [X21=0][X22=3][X23=0][X24=0][X25=0][X26=2]
         [X27=1][X28=0][X29=4][X30=0][X31=0][X32=0]
```

This solution is best according to heuristic S for the second level classes which split class 2.

```
          [X33=0][X34=0][X35=0]
          EVENTS COVERED: 11,18
 2    2   [X1=4..6][X2=0][X3=0][X4=1][X6=2..3][X7=3]          19          22
          [X8=1][X9=1][X10=0..1][X11=1][X12=1][X13=0]
          [X14=2][X15=2][X16=0][X17=0][X18=0][X19=1]
          [X20=0][X21=0][X22=3][X23=0][X24=0][X25=0]
          [X26=2][X27=1][X28=0][X29=4][X30=0][X31=0]
          [X32=0][X33=0][X34=0][X35=0]
          EVENTS COVERED: 17,19
 2    3   [X1=3..6][X2=0][X3=0][X4=2][X5=1][X6=0..2]          13          69
          [X7=2][X8=1][X9=0][X11=1][X12=1][X13=0][X14=2]
          [X15=2][X16=0][X17=0][X18=0][X19=1][X21=0]
          [X22=3][X23=0][X24=0][X25=0][X26=2][X27=1]
          [X28=0][X29=4][X30=0][X31=0][X32=0][X33=0]
          [X34=0][X35=0]
          EVENTS COVERED: 13,16,20
 2    4   [X1=3..6][X2=0][X3=0][X4=1..2][X5=0][X6=1]          14          30
          [X7=2,3][X8=1][X9=0,1][X10=0][X11=1][X12=1]
          [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]
          [X19=1][X20=0][X21=0][X22=3][X23=0][X24=0]
          [X25=0][X26=2][X27=1][X28=0][X29=4][X30=0]
          [X31=0][X32=0][X33=0][X34=0][X35=0]
          EVENTS COVERED: 12,14
 2        EXCEPTIONAL EVENTS: 15
 2    TOTALS                                                  127

(80 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  8.1E+003

WITH BETA=  3.00 THE BEST CLUSTERING AT THIS LEVEL IS FOR K=2          Class 2 is best subdivided into
                                                                       2 subclasses.
BEGINNING CLASSIFICATION BELOW HIERARCHY PATH 3-0

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


EXPERIMENT 1: K=2, CRITERION=F

THE 1 BEST CLUSTERINGS FOLLOW.     (2194 MS)

ITER/CLASS# VL-RULE                                          SEED --COSTS---
                                                                    SPAR
 2    1   [X1=3..6][X2=0][X3=2][X4=1][X6=1..3][X7=0]           1      378   This solution is best according
          [X8=1..2][X9=0,1][X10=1..2][X11=1][X12=1]                       to heuristic S for the second
          [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]                      level classes which split class
          [X19=1][X20=0][X21=3][X22=0,1][X23=1][X24=1]                    3.
          [X25=0][X26=0][X27=0][X28=0][X29=4][X30=0]
          [X31=0][X32=0][X33=0][X34=0][X35=0]
          EVENTS COVERED: 1,2,3,6,8,10
 2    2   [X1=3..6][X2=0][X3=2][X4=1][X5=0][X6=1..3]           5      284
          [X7=0,1][X8=1][X9=0,1][X11=1][X12=1][X13=0]
          [X14=2][X15=2][X16=0][X17=0][X18=0][X19=1]
          [X20=1][X21=3][X22=0,1][X23=1][X24=1][X25=0]
          [X26=0][X27=0][X28=0][X29=4][X30=0][X31=0]
          [X32=0][X33=0][X34=0][X35=0]
          EVENTS COVERED: 4,5,7,9
 2    TOTALS                                                  662

(12 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  5.3E+003

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


EXPERIMENT 1: K=3, CRITERION=F

THE 1 BEST CLUSTERINGS FOLLOW.     (5358 MS)
```

```
ITER/CLASS# VL-RULE                                                    SEED --COSTS---
                                                                              SPAR
 2    1  [X1=4..6][X2=0][X3=2][X4=1][X6=1  3][X7=0]                      1      140
         [X8=1..2][X9=0,1][X10=1..2][X11=1][X12=1]
         [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]
         [X19=1][X20=0][X21=3][X22=1][X23=1][X24=1]
         [X25=0][X26=0][X27=0][X28=0][X29=4][X30=0]
         [X31=0][X32=0][X33=0][X34=0][X35=0]
         EVENTS COVERED: 1,2,6,10
 2    2  [X1=5..6][X2=0][X3=2][X4=1][X5=0][X6=1..3]                      5      141
         [X7=0,1][X8=1][X9=0,1][X11=1][X12=1][X13=0]
         [X14=2][X15=2][X16=0][X17=0][X18=0][X19=1]
         [X20=1][X21=3][X22=0,1][X23=1][X24=1][X25=0]
         [X26=0][X27=0][X28=0][X29=4][X30=0][X31=0]
         [X32=0][X33=0][X34=0][X35=0]
         EVENTS COVERED: 5,7,9
 2    3  [X1=3][X2=0][X3=2][X4=1][X5=0][X6=1..2][X7=0,1]                 8       61
         [X8=1..2][X9=0,1][X10=1..2][X11=1][X12=1]
         [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]
         [X19=1][X21=3][X22=0][X23=1][X24=1][X25=0]
         [X26=0][X27=0][X28=0][X29=4][X30=0][X31=0]
         [X32=0][X33=0][X34=0][X35=0]
         EVENTS COVERED: 3,4,8
 2   TOTALS                                                                342

(30 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  9 2E+003

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


EXPERIMENT 1: K=4, CRITERION=F

THE 1 BEST CLUSTERINGS FOLLOW...   (7030 MS)

ITER/CLASS# VL-RULE                                                    SEED --COSTS---
                                                                              SPAR
 1    1  [X1=4..6][X2=0][X3=2][X4=1][X6=1][X7=0,1][X8=1]                 1       69
         [X9=0][X11=1][X12=1][X13=0][X14=2][X15=2]
         [X16=0][X17=0][X18=0][X19=1][X21=3][X22=1]
         [X23=1][X24=1][X25=0][X26=0][X27=0][X28=0]
         [X29=4][X30=0][X31=0][X32=0][X33=0][X34=0]
         [X35=0]
         EVENTS COVERED: 1,7,10
 1    2  [X1=5..6][X2=0][X3=2][X4=1][X5=0][X6=2  3]                      2       14
         [X7=0][X8=1][X9=1][X10=0  1][X11=1][X12=1]
         [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]
         [X19=1][X21=3][X22=1][X23=1][X24=1][X25=0]
         [X26=0][X27=0][X28=0][X29=4][X30=0][X31=0]
         [X32=0][X33=0][X34=0][X35=0]
         EVENTS COVERED: 2,5
 1    3  [X1=3  4][X2=0][X3=2][X4=1][X5=0][X6=1..3]                      3       45
         [X7=0][X8=2][X9=0,1][X10=1  2][X11=1][X12=1]
         [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]
         [X19=1][X20=0][X21=3][X22=0,1][X23=1][X24=1]
         [X25=0][X26=0][X27=0][X28=0][X29=4][X30=0]
         [X31=0][X32=0][X33=0][X34=0][X35=0]
         EVENTS COVERED: 3,6,8
 1    4  [X1=3..5][X2=0][X3=2][X4=1][X5=0][X6=2..3]                      4       22
         [X7=1][X8=1][X9=0,1][X10=1..2][X11=1][X12=1]
         [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]
         [X19=1][X20=1][X21=3][X22=0][X23=1][X24=1]
         [X25=0][X26=0][X27=0][X28=0][X29=4][X30=0]
         [X31=0][X32=0][X33=0][X34=0][X35=0]
         EVENTS COVERED: 4,9
 1   TOTALS                                                                150

(40 STARS BUILT)
```

```
FOR THE BEST SOLUTION ABOVE, S=  9.6E+003

WITH BETA=  3.00 THE BEST CLUSTERING AT THIS LEVEL IS FOR K=2

BEGINNING CLASSIFICATION BELOW HIERARCHY PATH 4-0

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1: K=2, CRITERION=F

THE 1 BEST CLUSTERINGS FOLLOW    (3988 MS)

ITER/CLASS# VL-RULE                                                SEED --COSTS---
                                                                          SPAR
 3    1  [X1=0..2][X2=1][X3=1..2][X4=1][X5=1][X6=1..3]              38      1722
         [X7=1,3][X8=1..2][X9=0,1][X11=1][X12=1][X13=0]
         [X14=2][X15=2][X16=0][X17=0][X18=0][X19=1]
         [X21=1,2][X22=2][X23=0][X24=1][X25=0][X26=0]
         [X27=0][X28=3][X29=4][X30=0][X31=0][X32=0]
         [X33=0][X34=0][X35=1]
         EVENTS COVERED: 36,38,41,43,45,46
 3    2  [X1=0..3][X2=1][X3=1..2][X4=0..1][X5=0][X7=1]              42      1525
         [X8=1..2][X9=0,1][X11=1][X12=1][X13=0][X14=2]
         [X15=2][X16=0][X17=0][X18=0][X19=1][X20=0]
         [X21=1,2][X22=2][X23=0][X24=0][X25=0][X26=0]
         [X27=0][X28=3][X29=4][X30=0][X31=0][X32=0]
         [X33=0][X34=0][X35=1]
         EVENTS COVERED: 31,32,33,34,35,37,39,40,42,44,47
 3  TOTALS                                                          3247

(15 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  2.6E+004

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1: K=3, CRITERION=F

THE 1 BEST CLUSTERINGS FOLLOW    (7678 MS)

ITER/CLASS# VL-RULE                                                SEED --COSTS---
                                                                          SPAR
 2    1  [X1=1..3][X2=1][X3=1..2][X4=0..1][X5=0][X7=1]              31       569
         [X8=2][X9=0,1][X11=1][X12=1][X13=0][X14=2]
         [X15=2][X16=0][X17=0][X18=0][X19=1][X20=0]
         [X21=1,2][X22=2][X23=0][X24=0][X25=0][X26=0]
         [X27=0][X28=3][X29=4][X30=0][X31=0][X32=0]
         [X33=0][X34=0][X35=1]
         EVENTS COVERED: 31,32,34,37,42,44,47
 2    2  [X1=0..2][X2=1][X3=1..2][X4=1][X5=1][X6=1..3]              36      1722
         [X7=1,3][X8=1..2][X9=0,1][X11=1][X12=1][X13=0]
         [X14=2][X15=2][X16=0][X17=0][X18=0][X19=1]
         [X21=1,2][X22=2][X23=0][X24=1][X25=0][X26=0]
         [X27=0][X28=3][X29=4][X30=0][X31=0][X32=0]
         [X33=0][X34=0][X35=1]
         EVENTS COVERED: 36,38,41,43,45,46
 2    3  [X1=0..1][X2=1][X3=1..2][X4=0..1][X5=0]                    33       284
         [X6=1..3][X7=1][X8=1][X9=0,1][X11=1][X12=1]
         [X13=0][X14=2][X15=2][X16=0][X17=0][X18=0]
         [X19=1][X20=0][X21=1,2][X22=2][X23=0][X24=0]
         [X25=0][X26=0][X27=0][X28=3][X29=4][X30=0]
         [X31=0][X32=0][X33=0][X34=0][X35=1]
         EVENTS COVERED: 33,35,39,40
 2  TOTALS                                                          2575

(24 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  7.0E+004
```

Class 3 is best subdivided into 2 subclasses

This solution is best according to heuristic S for the second level classes which split class 4.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1: K=4, CRITERION=F

THE 1 BEST CLUSTERINGS FOLLOW .     (23366 MS)

ITER/CLASS# VL-RULE                                                SEED --COSTS---
                                                                          SPAR
 5     1  [X1=1][X2=1][X3=2][X4=0..1][X6=2..3][X7=1,3]                 35      62
          [X8=1][X9=1][X10=1..2][X11=1][X12=1][X13=0]
          [X14=2][X15=2][X16=0][X17=0][X18=0][X19=1]
          [X20=0][X21=2][X22=2][X23=0][X24=0,1][X25=0]
          [X26=0][X27=0][X28=3][X29=4][X30=0][X31=0]
          [X32=0][X33=0][X34=0][X35=1]
          EVENTS COVERED: 35,43
 5     2  [X1=0..2][X2=1][X3=1..2][X4=1][X5=1][X6=1  3]                41     428
          [X7=1][X8=2][X9=0,1][X11=1][X12=1][X13=0]
          [X14=2][X15=2][X16=0][X17=0][X18=0][X19=1]
          [X21=1,2][X22=2][X23=0][X24=1][X25=0][X26=0]
          [X27=0][X28=3][X29=4][X30=0][X31=0][X32=0]
          [X33=0][X34=0][X35=1]
          EVENTS COVERED: 36,38,41,45
 5     3  [X1=0][X2=1][X3=1..2][X4=1][X6=1..3][X7=1]                   33      68
          [X8=1][X9=0][X11=1][X12=1][X13=0][X14=2][X15=2]
          [X16=0][X17=0][X18=0][X19=1][X20=0][X21=1]
          [X22=2][X23=0][X24=0,1][X25=0][X26=0][X27=0]
          [X28=3][X29=4][X30=0][X31=0][X32=0][X33=0]
          [X34=0][X35=1]
          EVENTS COVERED: 33,39,40,46
 5     4  [X1=1..3][X2=1][X3=1..2][X4=0  1][X5=0][X7=1]                37     569
          [X8=2][X9=0,1][X11=1][X12=1][X13=0][X14=2]
          [X15=2][X16=0][X17=0][X18=0][X19=1][X20=0]
          [X21=1,2][X22=2][X23=0][X24=0][X25=0][X26=0]
          [X27=0][X28=3][X29=4][X30=0][X31=0][X32=0]
          [X33=0][X34=0][X35=1]
          EVENTS COVERED: 31,32,34,37,42,44,47
 5   TOTALS                                                                1127

(70 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  7 2E+004

WITH BETA=  3.00 THE BEST CLUSTERING AT THIS LEVEL IS FOR K=2
```

Class 4 is best subdivided into 2 subclasses. Further development of the hierarchy is inhibited by the parameter MAX-HEIGHT set to 2, i.e., the hierarchy is to have only two levels.

```
ACTIVITY STATISTICS
CENDIST  =    94  CRITVAL  = 15939  CLUSTER  =    79
CMPCOV   =     0  EXTEND   =  5938  FREECPLX =  7925  GENRLIZ   = 11151
INTRSCT  = 12512  NID      =   313  NUMSEL   =     0  NEWCPLX   =  8671
REDUCE   =  5938  REFHIGH  = 34046  REFNUM   =     0  REFUNION  = 62042
REFLEV   =649959  REFLOW   =110394  SETMAP   = 15939  SETLEVELMA=     0
STAR     =   532  SEMANTICS =  1897  SYNDIST  =     0  TRIM      =  1307
TCOVER   =479364  DEGISCT  =     0  CLUSTERING=    5  TABLESETUP=     5
BESTC    =  3983  MCARD     = 26610  GENPATH   =  325  CLEARCV   =    30
SAVECV   =    79  STAR2     =     0
```

The procedure use counts are shown. The heaviest use is for REFLEV which is a function that counts the number of values in the value list part of a selector. TCOVER is a boolean function which test to see if one complex covers another. It is used heavily to determine which events are covered by a complex.

APPENDIX B

CLASSIFYING BLOCKS-WORLD FIGURES

B.1. Problem description

Given are 9 blocks-world figures each composed of 2 to four connected parts. The connections are denoted by the predicates ontop, inside, and next. Each part is described by the functions shape, size, and texture. The domain of the function shape is a hierarchy of values in which the shapes square, rectangle, diamond, and triangle generalize to the value polygon and the shapes circle and ellipse generalize to the value curved.

B.2. Input data file

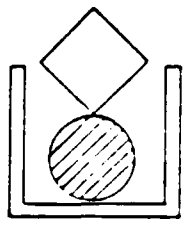
The following declarative statements are provided to INDUCE/3 to define this problem.

```
E
[SHAPE=SQUARE,RECTANGLE,DIAMOND,TRIANGLE] => [SHAPE=POLYGON].
E
[SHAPE=CIRCLE,ELLIPSE] => [SHAPE=CURVED].
```

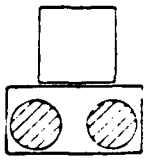
The E directive begins the definition of a generalized node in the value hierarchy in the domain of a function

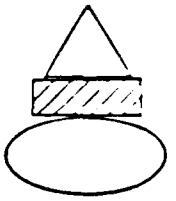
The M and A directives are used to introduce each object description. Each object has an associated decision category or class. The class assignments are not used by the clustering algorithm. The syntax is present for non-clustering applications such as building generalized rules which discriminate between multiple classes of objects.

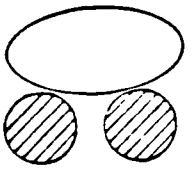
```
M
A
[ONTOP(P1,P2)][ONTOP(P2,P3)]
[SIZE(P1)=1][SIZE(P2)=1][SIZE(P3)=2]
[TEXTURE(P1)=0][TEXTURE(P2)=1][TEXTURE(P3)=0]
[SHAPE(P1)=DIAMOND][SHAPE(P2)=CIRCLE][SHAPE(P3)=USHAPE]
=>[D=1].
M
A
```

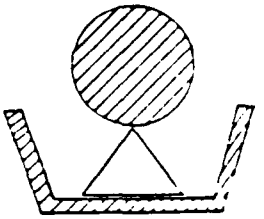


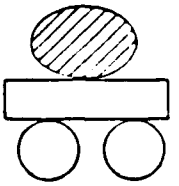
```
[ONTOP(P1,P2)]
[SIZE(P1)=1][SIZE(P2)=1][SIZE(P3)=0][SIZE(P4)=0]
[TEXTURE(P1)=0][TEXTURE(P2)=0][TEXTURE(P3)=1][TEXTURE(P4)=1]
[SHAPE(P1)=SQUARE][SHAPE(P2)=RECTANGLE]
[SHAPE(P3)=CIRCLE][SHAPE(P4)=CIRCLE]
=>[D=1].
M
A
[ONTOP(P1,P2)][ONTOP(P2,P3)]
[SIZE(P1)=1][SIZE(P2)=1][SIZE(P3)=2]
[TEXTURE(P1)=0][TEXTURE(P2)=1][TEXTURE(P3)=0]
[SHAPE(P1)=TRIANGLE][SHAPE(P2)=RECTANGLE][SHAPE(P3)=ELLIPSE]
=>[D=1].
M
A
[ONTOP(P1,P2)][ONTOP(P1,P3)]
[SIZE(P1)=2][SIZE(P2)=1][SIZE(P3)=1]
[TEXTURE(P1)=0][TEXTURE(P2)=1][TEXTURE(P3)=1]
[SHAPE(P1)=ELLIPSE][SHAPE(P2)=CIRCLE][SHAPE(P3)=CIRCLE]
=>[D=1].
M
A
[ONTOP(P1,P2)][ONTOP(P2,P3)]
[SIZE(P1)=2][SIZE(P2)=1][SIZE(P3)=2]
[TEXTURE(P1)=1][TEXTURE(P2)=0][TEXTURE(P3)=1]
[SHAPE(P1)=CIRCLE][SHAPE(P2)=TRIANGLE][SHAPE(P3)=USHAPE]
=>[D=1].
M
A
[ONTOP(P1,P2)][ONTOP(P2,P3)][ONTOP(P2,P4)]
[SIZE(P1)=1][SIZE(P2)=1][SIZE(P3)=0][SIZE(P4)=0]
[TEXTURE(P1)=1][TEXTURE(P2)=0][TEXTURE(P3)=0][TEXTURE(P4)=0]
[SHAPE(P1)=ELLIPSE][SHAPE(P2)=RECTANGLE]
[SHAPE(P3)=CIRCLE][SHAPE(P4)=CIRCLE]
=>[D=1].
M
A
[ONTOP(P1,P2)]
[SIZE(P1)=1][SIZE(P2)=1]
[TEXTURE(P1)=1][TEXTURE(P2)=1]
[SHAPE(P1)=TRIANGLE][SHAPE(P2)=ELLIPSE]
=>[D=1].
M
A
[ONTOP(P1,P2)]
[SIZE(P1)=2][SIZE(P2)=1]
[TEXTURE(P1)=1][TEXTURE(P2)=1]
[SHAPE(P1)=RECTANGLE][SHAPE(P2)=CIRCLE]
=>[D=1].
M
A
[ONTOP(P1,P2)][ONTOP(P2,P3)]
```

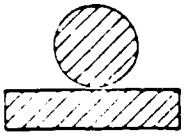




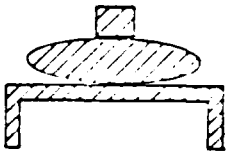








```
[SIZE(P1)=0..0][SIZE(P2)=2][SIZE(P3)=2]
[TEXTURE(P1)=1][TEXTURE(P2)=1][TEXTURE(P3)=1]
[SHAPE(P1)=SQUARE][SHAPE(P2)=ELLIPSE][SHAPE(P3)=USHAPE]
=>[D=1].
P
META 1
ALTER 4
VLMAXSTAR 4
AQMAXSTAR 4
NCONSIST 4
VLCRIT(1) -1
VLTOL(1) 0
VLCRIT(2) -2
VLTOL(2) 0
VLCRIT(3) 3
VLTOL(3) 0
EXTMTY
EQUIV
PA
QUIT
```



These statements assign values to control parameters. The parameter META controls the construction of zero-place functions such as the number of parts, the number of different values taken by a function, etc. VLMAXSTAR is the number of complexes maintained during star building. AQMAXSTAR is the number of complexes maintained during attributes-only star building. NCONSIST gives the number of alternative complexes constructed while building G(e|F) which have been specialized enough to cover no events in set F. The VLCRIT specifications give the order of application of elementary criteria in the LEF. By the specification "-1", the indication is to maximize (denoted by the minus sign) the value of criterion "1". This criterion measures the number of events covered. The second criterion maximizes the number of selectors in the complexes. The third criterion minimizes the number of events in F which are covered. The directive EXTMTY causes the object descriptions to be augmented with "MOST-" and "LEAST-" type predicates. The directive EQUIV causes the object descriptions to be augmented with "SAME-" type predicates. The PA directive prints out the control parameter values.

B.3. Results generated for k=2

```
INDUCE 3 - VERSION AS OF AUGUST 25 1984

THIS RUN MADE ON  84/09/18.
INITIALIZING...(MFIGS).....

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
VARIABLE NAME   VTYPE   VCOST          TRACE:
   #PT           LIN       1          STOPS:
   #P            LIN       1          PRINT RULES: TRUE
   #DIFF         LIN       1
   SHAPE         STR       1
   SIZE          LIN       1
```

This parameters block shows the values of various control parameters used by INDUCE/3. The function SHAPE has a structured domain. The function SIZE and the meta-variables #PT (number of parts of a certain type), #P (number of parts), and #DIFF (number of different values) all have linear domains. All other functions/predicates/variables have nominal domains.

- -

The control parameters fall

```
PARAMETERS  GENERAL          VL              AQ              L-RULE
        REGENSTAR    1    VLMAXSTAR  4    AQMAXSTAR  4    MAXL     5
        METATRIM     1    NCONSIST   4    AQCUTF1    20
        DESCTYPE  DISC    ALTER      4    LQST    FALSE
        EXTMTY    TRUE    MINCOVER 100
        EQUIV     TRUE    MAXBACK    0
```

into four groups general parameters (GENERAL heading), graph-representation heuristic control parameters (VL heading), attribute-representation heuristic control parameters (AQ heading), and background inference application parameters (L-RULE heading).

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
            VLNF=3              AQNF=2              LRNF=4
CRITERION  VLCRIT    VLTOL    AQCRIT    AQTOL    LRCRIT    LRTOL
             -1       0 00       -1      0 00       -1      0 00
             -2       0 00        2      0 00        2      0 00
              3       0 00                           -3      0 00
                                                      4      0 00
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

The program uses three LEFs, one for graph-represented rule evaluations (VLCRIT), one for subordinate value list generalization using the AQ procedure (AQCRIT), and one for inference rule evaluation (LRCRIT)

```
ENTER ONE CHAR: (P)ARAMETERS, (V)L1, (C)OVER, (M)ODIFY, (H)ELP FOR OPTIONS
? u

ENTER NUMBER OF SEEDS FOLLOWED BY THEIR EVENT NUMBERS
```

After the system has loaded the problem declarations, it becomes interactive with the user. User responses are in lower case. The u command requests the cluster operator.

```
? 2 1 9
NOW COVERING EVENT 1 AGAINST 9
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 5 4 3 2 1  (0 NEW)
RULE 23  EVENT SETS 1, COSTS(-6-2 3) -6 -1 0
[#PS(TEXTURE=0)#0]
```

In the present implementation, seed selection is performed by the user. The initial seeds are events 1 and 9. The first numeral is the value of k. The word RULE in the output listing means "complex."

*Each complex (RULE) is listed with a unique rule number (e.g., 23 for the above complex), a list by event number of events it covers (e.g., events 6 5 4 3 2 1), the number which are "new" (this is meaningless for clustering problems), and the scores (COSTS) on the evaluation criteria in the LEF (e.g., criteria are applied in the order (-)1 (-)2 3 which have scores -6, -1, and 0, respectively. Criterion 1 is to minimize the number of observed events covered. Criterion 2 is to minimize the number of selectors in the complex. Criterion 3 is to minimize the number of events covered in the against set F. A minus sign preceding the criterion number causes the use of the negated score. Thus the LEF used is to maximize the number of observed events covered by a complex (-1), then maximize the number of selectors in the complex (-2), and then minimize the number of events covered in set F. The relational operator # denotes "not equal", the symbol #PS denotes "number of object parts", the symbol #DIFF denotes "number of different values for", the symbol * denotes "any value"*

The stars G(1|9) and G(9|1) are built to determine generalized ways for covering each seed with conjunctive concepts which optimize the LEFs.

```
NOW COVERING EVENT 9 AGAINST 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7 5 4  (0 NEW)
RULE 43  EVENT SETS 1, COSTS(-6-2 3) -5 -1 0
[#PS(TEXTURE=0)=0..1]

EVENTS COVERED
 1 2 3 4 5 6
EVENTS COVERED
 4 5 7 8 9
MULTIPLY-COVERED EVENTS
 4 5
UNIQUELY DETERMINED
 1 2 3 6 7 8 9
NOW COVERING EVENT 1 AGAINST 9 8 7 5 4
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 3 2 1  (4 NEW)
RULE 194  EVENT SETS 1, COSTS(-6-2 3) -4 -41 0
```

The program analyzes the generated complexes to see if they are disjoint. Here, the complexes both cover events 4 and 5, so NID is performed to adjust the complexes to make them disjoint.

This is the beginning of the work done by NID. First, REFUNIONS are built for those events which were uniquely

covered

```
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [MST-ONTOP(P1)] [ONTOP(P1,P2)] [SAMESIZE(P1 P2)]
[SHAPE(P1)#RECTANGLE] [SHAPE(P2)#SQUARE] [SIZE(P1)=1] [SIZE(P2)=1] [TEXTURE(P1)=*]
[TEXTURE(P2)=*] [#DIFF-SHAPE=3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
 #PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1] [#PS(SHAPE=RECTANGLE)=0  1]
[#PS(SHAPE=DIAMOND)=0  1] [#PS(SHAPE=TRIANGLE)=0  1] [#PS(SHAPE=CIRCLE)#3]
[#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1] [#PS(SHAPE=POLYGON)=1  2]
[#PS(SIZE=1)=2] [#PS(SIZE=2)=0  1] [#PS(SIZE=0)#3] [#PS(TEXTURE=0)=2  3]
[#PS(TEXTURE=1)=1  2] [#PS=3  4]


NOW COVERING EVENT 9 AGAINST 6 5 4 3 2 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7  (3 NEW)
RULE 340  EVENT SETS 1, COSTS(-6-2 3) -3 -40 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE] [FORALL-PS(TEXTURE=1)]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SAMETEXTUR(P1 P2)] [SHAPE(P1)=POLYGON]
[SHAPE(P2)=CURVED] [SIZE(P1)=*] [SIZE(P2)#0] [TEXTURE(P1)=1] [TEXTURE(P2)=1]
[#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2] [#DIFF-TEXTURE=1] [#PS(SHAPE=POLYGON)=1]
[#PS(SHAPE=CURVED)=1] [#PS(SHAPE=SQUARE)=0  1] [#PS(SHAPE=RECTANGLE)=0  1]
[#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0  1] [#PS(SHAPE=CIRCLE)=0  1]
[#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1] [#PS(SIZE=1)#3] [#PS(SIZE=2)#3]
[#PS(SIZE=0)=0  1] [#PS(TEXTURE=0)=0] [#PS(TEXTURE=1)=2  3] [#PS=2  3]

UNCOVERED EVENTS 4 5
NOW COVERING EVENT 1 AGAINST 9 8 7 5
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 4 3 2 1  (5 NEW)
RULE 471  EVENT SETS 1, COSTS(-6-2 3) -5 -39 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE]
[FORALL-PS(TEXTURE=0)=FALSE] [FORALL-PS(TEXTURE=1)=FALSE] [MST-ONTOP(P1)]
[ONTOP(P1,P2)] [SHAPE(P1)#RECTANGLE] [SHAPE(P2)#SQUARE] [SIZE(P1)#0] [SIZE(P2)=1]
[TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=POLYGON)#3] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0  1] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SIZE=0)#3] [#PS(SIZE=1)=2] [#PS(SIZE=2)=0  1] [#PS(TEXTURE=1)=1  2]
[#PS(TEXTURE=0)#0] [#PS=3  4]

NOW COVERING EVENT 9 AGAINST 6 5 3 2 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7 4  (4 NEW)
RULE 602  EVENT SETS 1, COSTS(-6-2 3) -4 -37 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)=CIRCLE] [SHAPE(P2)=CURVED] [SIZE(P1)=*]
[SIZE(P2)#0] [TEXTURE(P2)=1] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)=0  1] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SIZE=2)#3] [#PS(SIZE=0)=0  1] [#PS(SIZE=1)#3]
```

Now NID tries to fit each uncovered event to a complex such that no event remains multiply covered. The result is a set of complexes which are disjoint or weakly intersecting. Here, event 4 is being fitted to each complex. It will remain in the complex which is the best host for it.

```
[#PS(TEXTURE=1)=2  3] [#PS(TEXTURE=0)=0  1] [#PS=2  3]

NOW COVERING EVENT 1 AGAINST 9 8 7 4
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 5 3 2 1  (5 NEW)
RULE 732: EVENT SETS 1, COSTS(-6-2 3) -5 -40 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)#RECTANGLE]
[SHAPE(P2)#SQUARE] [SIZE(P1)#0] [SIZE(P2)=1] [TEXTURE(P2)=*] [TEXTURE(P1)=*]
[#DIFF-SHAPE=3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2] [#PS(SHAPE=POLYGON)=1  2]
[#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1] [#PS(SHAPE=RECTANGLE)=0  1]
[#PS(SHAPE=DIAMOND)=0  1] [#PS(SHAPE=TRIANGLE)=0  1] [#PS(SHAPE=CIRCLE)#3]
[#PS(SHAPE=ELLIPSE)=0..1] [#PS(SHAPE=USHAPE)=0..1] [#PS(SIZE=0)#3] [#PS(SIZE=1)=1  2]
[#PS(SIZE=2)#3] [#PS(TEXTURE=1)=1  2] [#PS(TEXTURE=0)#0] [#PS=3..4]

NOW COVERING EVENT 9 AGAINST 6 3 2 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7 5 4  (5 NEW)
RULE 863: EVENT SETS 1, COSTS(-6-2 3) -5 -37 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)=*] [SHAPE(P2)#SQUARE] [SIZE(P1)=*]
[SIZE(P2)#0] [TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)=0..1] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SIZE=2)#3] [#PS(SIZE=0)=0..1] [#PS(SIZE=1)#3]
[#PS(TEXTURE=1)=2..3] [#PS(TEXTURE=0)=0..1] [#PS=2..3]

RULE COVERING CLUSTER 1  (EVENTS 6 3 2 1)
RULE 194: EVENT SETS 1, COSTS(-6-2 3) -4 -41 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE]
[FORALL-PS(TEXTURE=0)=FALSE] [FORALL-PS(TEXTURE=1)=FALSE] [MST-ONTOP(P1)]
[ONTOP(P1,P2)] [SAMESIZE(P1 P2)] [SHAPE(P1)#RECTANGLE] [SHAPE(P2)#SQUARE]
[SIZE(P1)=1] [SIZE(P2)=1] [TEXTURE(P1)=*] [TEXTURE(P2)=*] [#DIFF-SHAPE=3]
[#DIFF-SIZE=2] [#DIFF-TEXTURE=2] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0..1]
[#PS(SHAPE=RECTANGLE)=0..1] [#PS(SHAPE=DIAMOND)=0..1] [#PS(SHAPE=TRIANGLE)=0..1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SHAPE=ELLIPSE)=0..1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)=1..2] [#PS(SIZE=1)=2] [#PS(SIZE=2)=0..1] [#PS(SIZE=0)#3]
[#PS(TEXTURE=0)=2..3] [#PS(TEXTURE=1)=1..2] [#PS=3..4]

RULE COVERING CLUSTER 2  (EVENTS 9 8 7 5 4)
RULE 863: EVENT SETS 1, COSTS(-6-2 3) -5 -37 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)=*] [SHAPE(P2)#SQUARE] [SIZE(P1)=*]
[SIZE(P2)#0] [TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)=0  1] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SIZE=2)#3] [#PS(SIZE=0)=0  1] [#PS(SIZE=1)#3]
[#PS(TEXTURE=1)=2  3] [#PS(TEXTURE=0)=0  1] [#PS=2  3]
```

Now event 5 is fitted to a complex.

Now that each multiply covered event has been fitted to a complex, NID is finished. The complete descriptions of the clusters are given here. An edited description comprised of the descriptors which discriminate between clusters is shown in Fig. 6.5 (upper figure).

```
THE FOLLOWING EVENTS COULD NOT BE COVERED:  (NONE)
```

All events were covered by NID

```
FINDING NEW SEEDS
CNO=1 EVENT=1 DSUM=  4 2000000000000E+001
CNO=1 EVENT=2 DSUM=  4 2000000000000E+001
CNO=1 EVENT=3 DSUM=  3 6000000000000E+001
CNO=1 EVENT=6 DSUM=  4 0000000000000E+001
CNO=2 EVENT=4 DSUM=  7 5000000000000E+001
CNO=2 EVENT=4 DSUM=  7 5000000000000E+001
CNO=2 EVENT=5 DSUM=  9 2000000000000E+001
CNO=2 EVENT=7 DSUM=  8 0000000000000E+001
CNO=2 EVENT=8 DSUM=  7 5000000000000E+001
CNO=2 EVENT=9 DSUM=  9 5000000000000E+001
```

The following table shows the values of sum of (syntactic) distance scores for each event. The distance measure is a modification of syntactic distance described in Chapter 5 applied to the attribute values generated by performing template matching of the cluster description with the events. Note that the cluster description matches event 4 in two ways. CNO indicates the cluster number for the event. The event in each cluster with the minimum DSUM value is used as the seed event in the next iteration.

```
ENTER NUMBER OF SEEDS FOLLOWED BY THEIR EVENT NUMBERS
? 2 3 8
NOW COVERING EVENT 3 AGAINST 8
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 5 4 3 2 1  (0 NEW)
RULE 884  EVENT SETS 1, COSTS(-6-2 3) -6 -1 0
[#PS(TEXTURE=0)#0]

NOW COVERING EVENT 8 AGAINST 3
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7 5 4 2  (0 NEW)
RULE 899  EVENT SETS 1, COSTS(-6-2 3) -6 -1 0
[#PS(TEXTURE=1)=2  3]

EVENTS COVERED
 1 2 3 4 5 6
EVENTS COVERED
 2 4 5 7 8 9
MULTIPLY-COVERED EVENTS
 2 4 5
UNIQUELY DETERMINED
 1 3 6 7 8 9
NOW COVERING EVENT 3 AGAINST 9 8 7 5 4 2
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 3 1  (3 NEW)
RULE 1139  EVENT SETS 1, COSTS(-6-2 3) -3 -46 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [LST-ONTOP(P3)] [MST-ONTOP(P1)] [ONTOP(P1,P2)]
[ONTOP(P2,P3)] [SAMESIZE(P1,P2)] [SHAPE(P1)#SQUARE] [SHAPE(P2)#SQUARE]
[SHAPE(P3)#POLYGON] [SIZE(P1)=1] [SIZE(P2)=1] [SIZE(P3)=*] [TEXTURE(P1)=*]
[TEXTURE(P2)=*] [TEXTURE(P3)=0] [#DIFF-SHAPE=3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=CIRCLE)#3] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)=1  2] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0  1] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SIZE=0)#3] [#PS(SIZE=1)=2] [#PS(SIZE=2)=0  1] [#PS(TEXTURE=1)=1]
[#PS(TEXTURE=0)=2  3] [#PS=3  4]

NOW COVERING EVENT 8 AGAINST 6 5 4 3 2 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7  (3 NEW)
RULE 1287  EVENT SETS 1, COSTS(-6-2 3) -3 -40 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
```

Events 9 and 8 have the minimum DSUM score (there is a tie between events 8 and 4) and are selected as seeds for the next iteration.

```
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE] [FORALL-PS(TEXTURE=1)]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SAMETEXTUR(P1 P2)] [SHAPE(P1)=POLYGON]
[SHAPE(P2)=CURVED] [SIZE(P1)=*] [SIZE(P2)#0] [TEXTURE(P1)=1] [TEXTURE(P2)=1]
[#DIFF-SHAPE=2 .3] [#DIFF-SIZE=1 .2] [#DIFF-TEXTURE=1] [#PS(SHAPE=POLYGON)=1]
[#PS(SHAPE=CURVED)=1] [#PS(SHAPE=SQUARE)=0 1] [#PS(SHAPE=RECTANGLE)=0 1]
[#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0 1] [#PS(SHAPE=CIRCLE)=0 1]
[#PS(SHAPE=ELLIPSE)=0 1] [#PS(SHAPE=USHAPE)=0 1] [#PS(SIZE=1)#3] [#PS(SIZE=2)#3]
[#PS(SIZE=0)=0..1] [#PS(TEXTURE=0)=0] [#PS(TEXTURE=1)=2 3] [#PS=2 3]

UNCOVERED EVENTS 2 4 5
NOW COVERING EVENT 3 AGAINST 9 8 7 5 4
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 3 2 1 (4 NEW)
RULE 1434 EVENT SETS 1, COSTS(-6-2 3) -4 -41 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [MST-ONTOP(P1)] [ONTOP(P1,P2)] [SAMESIZE(P1 P2)]
[SHAPE(P1)#RECTANGLE] [SHAPE(P2)#SQUARE] [SIZE(P1)=1] [SIZE(P2)=1] [TEXTURE(P1)=*]
[TEXTURE(P2)=*] [#DIFF-SHAPE=3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0 1] [#PS(SHAPE=RECTANGLE)=0 1]
[#PS(SHAPE=DIAMOND)=0 1] [#PS(SHAPE=TRIANGLE)=0 1] [#PS(SHAPE=CIRCLE)#3]
[#PS(SHAPE=ELLIPSE)=0 1] [#PS(SHAPE=USHAPE)=0 1] [#PS(SHAPE=POLYGON)=1 2]
[#PS(SIZE=1)=2] [#PS(SIZE=2)=0 1] [#PS(SIZE=0)#3] [#PS(TEXTURE=0)=2 3]
[#PS(TEXTURE=1)=1 2] [#PS=3 4]

NOW COVERING EVENT 8 AGAINST 6 5 4 3 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7 2 (4 NEW)
RULE 1580 EVENT SETS 1, COSTS(-6-2 3) -4 -39 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
 ]RALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
 ORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE] [MST-ONTOP(P1)]
[ONTOP(P1,P2)] [SAMETEXTUR(P1 P2)] [SHAPE(P1)=POLYGON] [SHAPE(P2)#SQUARE]
[SIZE(P1)=*] [SIZE(P2)#0] [TEXTURE(P1)=*] [TEXTURE(P2)=*] [#DIFF-SHAPE=2 3]
[#DIFF-SIZE=1..2] [#DIFF-TEXTURE=1..2] [#PS(SHAPE=USHAPE)=0 1]
[#PS(SHAPE=POLYGON)=1 2] [#PS(SHAPE=CURVED)=1 2] [#PS(SHAPE=SQUARE)=0 1]
[#PS(SHAPE=RECTANGLE)=0 1] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0 1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SHAPE=ELLIPSE)=0..1] [#PS(SIZE=0)#3] [#PS(SIZE=1)#3]
[#PS(SIZE=2)#3] [#PS(TEXTURE=0)#3] [#PS(TEXTURE=1)=2..3] [#PS#0..1]

NOW COVERING EVENT 3 AGAINST 9 8 7 5 2
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 4 3 1 (4 NEW)
RULE 1711 EVENT SETS 1, COSTS(-6-2 3) -4 -39 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE]
[FORALL-PS(TEXTURE=0)=FALSE] [FORALL-PS(TEXTURE=1)=FALSE] [MST-ONTOP(P1)]
[ONTOP(P1,P2)] [SHAPE(P1)#SQUARE] [SHAPE(P2)#SQUARE] [SIZE(P1)#0] [SIZE(P2)=1]
[TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2..3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=POLYGON)#3] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0]
[#PS(SHAPE=RECTANGLE)=0 1] [#PS(SHAPE=DIAMOND)=0..1] [#PS(SHAPE=TRIANGLE)=0 1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SHAPE=ELLIPSE)=0 1] [#PS(SHAPE=USHAPE)=0 1]
[#PS(SIZE=0)#3] [#PS(SIZE=1)=2] [#PS(SIZE=2)=0..1] [#PS(TEXTURE=1)=1 2]
[#PS(TEXTURE=0)#0] [#PS=3 4]
```

```
NOW COVERING EVENT 8 AGAINST 6 5 3 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7 4 2  (5 NEW)
RULE 1842  EVENT SETS 1, COSTS(-6-2 3) -5 -37 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)#CIRCLE] [SHAPE(P2)#SQUARE] [SIZE(P1)=*]
[SIZE(P2)#0] [TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)#3] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SIZE=2)#3] [#PS(SIZE=0)#3] [#PS(SIZE=1)#3]
[#PS(TEXTURE=1)=2  3] [#PS(TEXTURE=0)#3] [#PS#0  1]

NOW COVERING EVENT 3 AGAINST 9 8 7 4 2
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 5 3 1  (4 NEW)
RULE 2058  EVENT SETS 1, COSTS(-6-2 3) -4 -45 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [LST-ONTOP(P3)] [MST-ONTOP(P1)] [ONTOP(P1,P2)]
[ONTOP(P2,P3)] [SHAPE(P1)#SQUARE] [SHAPE(P2)#SQUARE] [SHAPE(P3)#POLYGON]
[SIZE(P1)#0] [SIZE(P2)=1] [SIZE(P3)=*] [TEXTURE(P3)=*] [TEXTURE(P1)=*]
[TEXTURE(P2)=*] [#DIFF-SHAPE=3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=TRIANGLE)=0  1] [#PS(SHAPE=CIRCLE)#3] [#PS(SHAPE=ELLIPSE)=0  1]
[#PS(SHAPE=USHAPE)=0  1] [#PS(SHAPE=POLYGON)=1  2] [#PS(SHAPE=CURVED)#0]
[#PS(SHAPE=SQUARE)=0] [#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0  1]
[#PS(SIZE=2)#3] [#PS(SIZE=0)#3] [#PS(SIZE=1)=1  2] [#PS(TEXTURE=0)#0]
[#PS(TEXTURE=1)=1  2] [#PS=3  4]

NOW COVERING EVENT 8 AGAINST 6 3 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7 5 4 2  (6 NEW)
RULE 2192  EVENT SETS 1, COSTS(-6-2 3) -6 -37 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)=*] [SHAPE(P2)#SQUARE] [SIZE(P1)=*]
[SIZE(P2)#0] [TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)#3] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SIZE=2)#3] [#PS(SIZE=0)#3] [#PS(SIZE=1)#3]
[#PS(TEXTURE=1)=2  3] [#PS(TEXTURE=0)#3] [#PS#0  1]

RULE COVERING CLUSTER 1   (EVENTS 6 3 1)
RULE 1139  EVENT SETS 1, COSTS(-6-2 3) -3 -46 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [LST-ONTOP(P3)] [MST-ONTOP(P1)] [ONTOP(P1,P2)]
[ONTOP(P2,P3)] [SAMESIZE(P1 P2)] [SHAPE(P1)#SQUARE] [SHAPE(P2)#SQUARE]
[SHAPE(P3)#POLYGON] [SIZE(P1)=1] [SIZE(P2)=1] [SIZE(P3)=*] [TEXTURE(P1)=*]
[TEXTURE(P2)=*] [TEXTURE(P3)=0] [#DIFF-SHAPE=3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=CIRCLE)#3] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)=1  2] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0  1] [#PS(SHAPE=TRIANGLE)=0  1]
```

The resulting cluster descriptions are given here. These descriptions, edited to reveal the descriptors which discriminate between classes, are given in Fig 6 5 (lower figure)

```
[#PS(SIZE=0)#3] [#PS(SIZE=1)=2] [#PS(SIZE=2)=0  1] [#PS(TEXTURE=1)=1]
[#PS(TEXTURE=0)=2  3] [#PS=3  4]

RULE COVERING CLUSTER 2   (EVENTS 9 8 7 5 4 2)
RULE 2192  EVENT SETS 1, COSTS(-6-2 3) -6 -37 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)=*] [SHAPE(P2)#SQUARE] [SIZE(P1)=*]
[SIZE(P2)#0] [TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)#3] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SIZE=2)#3] [#PS(SIZE=0)#3] [#PS(SIZE=1)#3]
[#PS(TEXTURE=1)=2  3] [#PS(TEXTURE=0)#3] [#PS#0  1]

THE FOLLOWING EVENTS COULD NOT BE COVERED:  (NONE)
FINDING NEWSEEDS
CNO=1 EVENT=1 DSUM=  5.2000000000000E+001
CNO=1 EVENT=3 DSUM=  4.6000000000000E+001
CNO=1 EVENT=6 DSUM=  3.9000000000000E+001
CNO=1 EVENT=6 DSUM=  3.9000000000000E+001
CNO=2 EVENT=2 DSUM=  1.1200000000000E+002
CNO=2 EVENT=4 DSUM=  9.3000000000000E+001
CNO=2 EVENT=4 DSUM=  9.3000000000000E+001
CNO=2 EVENT=5 DSUM=  1.1200000000000E+002
CNO=2 EVENT=7 DSUM=  9.7000000000000E+001
CNO=2 EVENT=8 DSUM=  9.3000000000000E+001
CNO=2 EVENT=9 DSUM=  1.1600000000000E+002

ENTER NUMBER OF SEEDS FOLLOWED BY THEIR EVENT NUMBERS
? 2 6 4
NOW COVERING EVENT 6 AGAINST 4
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 3 2 1  (0 NEW)
RULE 2212  EVENT SETS 1, COSTS(-6-2 3) -4 -1 0
[#PS(TEXTURE=0)=2  3]

NOW COVERING EVENT 4 AGAINST 6
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7 5 4 3 2 1  (0 NEW)
RULE 2233  EVENT SETS 1, COSTS(-6-2 3) -8 -1 0
[#PS(TEXTURE=0)#3]

EVENTS COVERED
 1 2 3 6
EVENTS COVERED
 1 2 3 4 5 7 8 9
MULTIPLY-COVERED EVENTS
 1 2 3
UNIQUELY DETERMINED
 4 5 6 7 8 9
NOW COVERING EVENT 6 AGAINST 9 8 7 5 4 3 2 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6  (1 NEW)
RULE 2573  EVENT SETS 1, COSTS(-6-2 3) -1 -54 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [LST-ONTOP(P3)] [LST-ONTOP(P4)] [MST-ONTOP(P1)]
[ONTOP(P1,P2)] [ONTOP(P2,P4)] [ONTOP(P2,P3)] [SAMESHAPE(P3 P4)] [SAMESIZE(P3 P4)]
[SAMESIZE(P1 P2)] [SAMETEXTUR(P2 P3 P4)] [SHAPE(P1)=ELLIPSE] [SHAPE(P2)=RECTANGLE]
[SHAPE(P3)=CIRCLE] [SHAPE(P4)=CIRCLE] [SIZE(P1)=1] [SIZE(P2)=1] [SIZE(P3)=0]
```

NID was able to cover all events.

The next choice of seed events is events 6 and 4.

```
[SIZE(P4)=0] [TEXTURE(P4)=0] [TEXTURE(P1)=1] [TEXTURE(P2)=0] [TEXTURE(P3)=0]
[#DIFF-SHAPE-3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2] [#PS(SHAPE=TRIANGLE)=0]
[#PS(SHAPE=CIRCLE)=2] [#PS(SHAPE=ELLIPSE)=1] [#PS(SHAPE=USHAPE)=0]
[#PS(SHAPE=POLYGON)=1] [#PS(SHAPE=CURVED)=3] [#PS(SHAPE=SQUARE)=0]
[#PS(SHAPE=RECTANGLE)=1] [#PS(SHAPE=DIAMOND)=0] [#PS(SIZE=2)=0] [#PS(SIZE=0)=2]
[#PS(SIZE=1)=2] [#PS(TEXTURE=1)=1] [#PS(TEXTURE=0)=3] [#PS=4]

NOW COVERING EVENT 4 AGAINST 6 3 2 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7 5 4  (5 NEW)
RULE 2697  EVENT SETS 1, COSTS(-6-2 3) -5 -37 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)=*] [SHAPE(P2)#SQUARE] [SIZE(P1)=*]
[SIZE(P2)#0] [TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)=0  1] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SIZE=2)#3] [#PS(SIZE=0)=0  1] [#PS(SIZE=1)#3]
[#PS(TEXTURE=1)=2  3] [#PS(TEXTURE=0)=0  1] [#PS=2  3]

UNCOVERED EVENTS 1 2 3
NOW COVERING EVENT 6 AGAINST 9 8 7 5 4 3 2
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 1  (2 NEW)
RULE 2934  EVENT SETS 1, COSTS(-6-2 3) -2 -46 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [LST-ONTOP(P3)] [MST-ONTOP(P1)] [ONTOP(P1,P2)]
[ONTOP(P2,P3)] [SAMESIZE(P1 P2)] [SHAPE(P1)#SQUARE] [SHAPE(P2)#SQUARE]
[SHAPE(P3)#POLYGON] [SIZE(P1)=1] [SIZE(P2)=1] [SIZE(P3)=*] [TEXTURE(P1)=*]
[TEXTURE(P2)=*] [TEXTURE(P3)=0] [#DIFF-SHAPE=3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=CIRCLE)=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)=1] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0  1] [#PS(SHAPE=TRIANGLE)=0]
[#PS(SIZE=0)#3] [#PS(SIZE=1)=2] [#PS(SIZE=2)=0  1] [#PS(TEXTURE=1)=1]
[#PS(TEXTURE=0)=2  3] [#PS=3  4]

NOW COVERING EVENT 4 AGAINST 6 3 2
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7 5 4 1  (6 NEW)
RULE 3066  EVENT SETS 1, COSTS(-6-2 3) -6 -37 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)=*] [SHAPE(P2)#SQUARE] [SIZE(P1)=*]
[SIZE(P2)#0] [TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)=0  1] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0  1] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SIZE=2)#3] [#PS(SIZE=0)=0  1] [#PS(SIZE=1)#3]
[#PS(TEXTURE=1)#0] [#PS(TEXTURE=0)#3] [#PS=2  3]

NOW COVERING EVENT 6 AGAINST 9 8 7 5 4 3 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 2  (2 NEW)
RULE 3212  EVENT SETS 1, COSTS(-6-2 3) -2 -41 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
```

```
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [MST-ONTOP(P1)] [ONTOP(P1,P2)] [SAMESIZE(P1 P2)]
[SHAPE(P1)#RECTANGLE] [SHAPE(P2)=RECTANGLE] [SIZE(P1)=1] [SIZE(P2)=1]
[TEXTURE(P1)=*] [TEXTURE(P2)=0] [#DIFF-SHAPE=3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=CURVED)=2..3] [#PS(SHAPE=SQUARE)=0..1] [#PS(SHAPE=RECTANGLE)=1]
[#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0] [#PS(SHAPE=CIRCLE)=2]
[#PS(SHAPE=ELLIPSE)=0..1] [#PS(SHAPE=USHAPE)=0] [#PS(SHAPE=POLYGON)=1..2]
[#PS(SIZE=1)=2] [#PS(SIZE=2)=0] [#PS(SIZE=0)=2] [#PS(TEXTURE=0)=2..3]
[#PS(TEXTURE=1)=1..2] [#PS=4]

NOW COVERING EVENT 4 AGAINST 6 3
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7 5 4 3 2 1  (7 NEW)
RULE 3342  EVENT SETS 1, COSTS(-6-2 3) -7 -37 1
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)=*] [SHAPE(P2)#SQUARE] [SIZE(P1)=*]
[SIZE(P2)#0] [TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2..3] [#DIFF-SIZE=1..2]
[#DIFF-TEXTURE=1..2] [#PS(SHAPE=ELLIPSE)=0..1] [#PS(SHAPE=USHAPE)=0..1]
[#PS(SHAPE=POLYGON)#3] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0..1]
[#PS(SHAPE=RECTANGLE)=0..1] [#PS(SHAPE=DIAMOND)=0..1] [#PS(SHAPE=TRIANGLE)=0..1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SIZE=2)#3] [#PS(SIZE=0)#3] [#PS(SIZE=1)#3]
[#PS(TEXTURE=1)#0] [#PS(TEXTURE=0)#3] [#PS#0..1]

NOW COVERING EVENT 6 AGAINST 9 8 7 5 4 2 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 3  (2 NEW)
RULE 3579  EVENT SETS 1, COSTS(-6-2 3) -2 -46 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [LST-ONTOP(P3)] [MST-ONTOP(P1)] [ONTOP(P1,P2)]
[ONTOP(P2,P3)] [SAMESIZE(P1 P2)] [SHAPE(P1)#SQUARE] [SHAPE(P2)=RECTANGLE]
[SHAPE(P3)=CURVED] [SIZE(P1)=1] [SIZE(P2)=1] [SIZE(P3)=*] [TEXTURE(P1)=*]
[TEXTURE(P2)=*] [TEXTURE(P3)=0] [#DIFF-SHAPE=3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=CIRCLE)#3] [#PS(SHAPE=ELLIPSE)=1] [#PS(SHAPE=USHAPE)=0]
[#PS(SHAPE=POLYGON)=1..2] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0]
[#PS(SHAPE=RECTANGLE)=1] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0..1]
[#PS(SIZE=0)#3] [#PS(SIZE=1)=2] [#PS(SIZE=2)=0..1] [#PS(TEXTURE=1)=1]
[#PS(TEXTURE=0)=2..3] [#PS=3..4]

NOW COVERING EVENT 4 AGAINST 6
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7 5 4 3 2 1  (8 NEW)
RULE 3711  EVENT SETS 1, COSTS(-6-2 3) -8 -37 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)=*] [SHAPE(P2)#SQUARE] [SIZE(P1)=*]
[SIZE(P2)#0] [TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2..3] [#DIFF-SIZE=1..2]
[#DIFF-TEXTURE=1..2] [#PS(SHAPE=ELLIPSE)=0..1] [#PS(SHAPE=USHAPE)=0..1]
[#PS(SHAPE=POLYGON)#3] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0..1]
[#PS(SHAPE=RECTANGLE)=0..1] [#PS(SHAPE=DIAMOND)=0..1] [#PS(SHAPE=TRIANGLE)=0..1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SIZE=2)#3] [#PS(SIZE=0)#3] [#PS(SIZE=1)#3]
[#PS(TEXTURE=1)#0] [#PS(TEXTURE=0)#3] [#PS#0..1]

RULE COVERING CLUSTER 1.  (EVENTS 6)
RULE 2573  EVENT SETS 1, COSTS(-6-2 3) -1 -54 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
```

```
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [LST-ONTOP(P3)] [LST-ONTOP(P4)] [MST-ONTOP(P1)]
[ONTOP(P1,P2)] [ONTOP(P2,P4)] [ONTOP(P2,P3)] [SAMESHAPE(P3 P4)] [SAMESIZE(P3 P4)]
[SAMESIZE(P1 P2)] [SAMETEXTUR(P2 P3 P4)] [SHAPE(P1)=ELLIPSE] [SHAPE(P2)=RECTANGLE]
[SHAPE(P3)=CIRCLE] [SHAPE(P4)=CIRCLE] [SIZE(P1)=1] [SIZE(P2)=1] [SIZE(P3)=0]
[SIZE(P4)=0] [TEXTURE(P4)=0] [TEXTURE(P1)=1] [TEXTURE(P2)=0] [TEXTURE(P3)=0]
[#DIFF-SHAPE=3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2] [#PS(SHAPE=TRIANGLE)=0]
[#PS(SHAPE=CIRCLE)=2] [#PS(SHAPE=ELLIPSE)=1] [#PS(SHAPE=USHAPE)=0]
[#PS(SHAPE=POLYGON)=1] [#PS(SHAPE=CURVED)=3] [#PS(SHAPE=SQUARE)=0]
[#PS(SHAPE=RECTANGLE)=1] [#PS(SHAPE=DIAMOND)=0] [#PS(SIZE=2)=0] [#PS(SIZE=0)=2]
[#PS(SIZE=1)=2] [#PS(TEXTURE=1)=1] [#PS(TEXTURE=0)=3] [#PS=4]


RULE COVERING CLUSTER 2   (EVENTS 9 8 7 5 4 3 2 1)
RULE 3711  EVENT SETS 1, COSTS(-6-2 3) -8 -37 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)=*] [SHAPE(P2)#SQUARE] [SIZE(P1)=*]
[SIZE(P2)#0] [TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)#3] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0  1] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SIZE=2)#3] [#PS(SIZE=0)#3] [#PS(SIZE=1)#3]
[#PS(TEXTURE=1)#0] [#PS(TEXTURE=0)#3] [#PS#0  1]


THE FOLLOWING EVENTS COULD NOT BE COVERED   (NONE)
FINDING NEWSEEDS
CNO=1 EVENT=6 DSUM=  0 00000000000000E+000
CNO=1 EVENT=6 DSUM=  0 00000000000000E+000
CNO=2 EVENT=1 DSUM=  1 26000000000000E+002
CNO=2 EVENT=2 DSUM=  1 43000000000000E+002
CNO=2 EVENT=3 DSUM=  1 25000000000000E+002
CNO=2 EVENT=4 DSUM=  1 29000000000000E+002
CNO=2 EVENT=4 DSUM=  1 29000000000000E+002
CNO=2 EVENT=5 DSUM=  1 39000000000000E+002
CNO=2 EVENT=7 DSUM=  1 29000000000000E+002
CNO=2 EVENT=8 DSUM=  1 29000000000000E+002
CNO=2 EVENT=9 DSUM=  1 51000000000000E-002

ENTER ONE CHAR  (P)ARAMETERS, (V)L1, (C)OVER, (M)ODIFY, (H)ELP FOR OPTIONS
? q
CPU TIME USED    169 321 SEC

TABLE OF PROCEDURE USE COUNTS
    352-ALLC            50-AQ             72-AQSET           9-ADDSEL
      0-ADDCONS        281-ABSOURB         0-APPLRULES       0-ADDLRULES
    314-ADDLINK       1978-COSTC           0-CLEANCPX        3-CLUSTER
     28-CCOVER        8365-COSTG          28-COVER           0-COSTL
      0-CALCARITH      896-COMPMS         790-EXTND        10084-FREETAB
   3779-FREEGRPH      4900-FREECPX          0-SUBGDETAIL     92-LQST2
      0-LSCHOOSE       28-ADDMETA          72-ADDTOMQ       444-NEWCPXTAB
   5001-NEWC          3789-NEWG           810-NEWGP          3-NEWSEEDS
   3850-COPYCPX       3850-COPYGRPH         0-PROCARITH    62268-SUBG1
  29462-SUBG           144-TRIMC          253-TRIMG          0-TRIML
     28-TRIMM          810-TOOSMALL         0-VL1           11-VLINT
  11074-CPXCOV         383-GENRLIZ

                ARITH     SGWA    LRULE    CPXTB     CPX    GRAPH
 NEW COUNTS         0        1        0       10      38       37
NUMBER FREE         -        -        -       10      28       27
```

The DSUM values are much worse than the previous iteration. This marked decline in cohesiveness is used as the stopping criterion

The NEW COUNTS show the maximum number of data structures needed for the problem. The structure called

B.4. Results generated for k=3

GRAPH holds a complex in graph form. The structure CPX holds a vector of attributes used in the selector value list generalization phase. Although 2911 different complexes were generated, only 37 were ever stored simultaneously.

```
INDUCE 3 - VERSION AS OF AUGUST 25 1984

THIS RUN MADE ON  84/09/18
INITIALIZING   (MFIGS)

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
VARIABLE NAME   VTYPE   VCOST          TRACE
   #PT           LIN       1          STOPS
   #P            LIN       1          PRINT RULES  TRUE
   #DIFF         LIN       1
   SHAPE         STR       1
   SIZE          LIN       1

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PARAMETERS  GENERAL          VL              AQ            L-RULE
        REGENSTAR    1    VLMAXSTAR  4    AQMAXSTAR  4    MAXL     5
        METATRIM     1    NCONSIST   4    AQCUTF1   20
        DESCTYPE  DISC    ALTER      4    LQST   FALSE
        EXTMTY    TRUE    MINCOVER 100
        EQUIV     TRUE    MAXBACK    0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
            VLNF=3              AQNF=2              LRNF=4
CRITERION  VLCRIT    VLTOL    AQCRIT    AQTOL    LRCRIT    LRTOL
             -1       0 00       -1       0 00      -1       0 00
             -2       0 00        2       0 00       2       0 00
              3       0 00                           -3       0 00
                                                      4       0 00
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

ENTER ONE CHAR  (P)ARAMETERS, (V)L1, (C)OVER, (M)ODIFY, (H)ELP FOR OPTIONS
? u

ENTER NUMBER OF SEEDS FOLLOWED BY THEIR EVENT NUMBERS
? 3 1 5 9
NOW COVERING EVENT 1 AGAINST 9 5
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 8 7 6 4 3 2 1  (0 NEW)
RULE 25  EVENT SETS 1, COSTS(-6-2 3) -7 -1 0
[#PS(SIZE=2)=0  1]

NOW COVERING EVENT 5 AGAINST 9 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 5 4  (0 NEW)
RULE 45  EVENT SETS 1, COSTS(-6-2 3) -2 -1 0
[#PS(TEXTURE=0)=1]

NOW COVERING EVENT 9 AGAINST 5 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7  (0 NEW)
RULE 65  EVENT SETS 1, COSTS(-6-2 3) -3 -1 0
[#PS(TEXTURE=0)=0]

EVENTS COVERED
 1 2 3 4 6 7 8
EVENTS COVERED
 4 5
EVENTS COVERED
 7 8 9
MULTIPLY-COVERED EVENTS
 4 7 8
UNIQUELY DETERMINED
```

The response directs the construction of three classes. The initial seeds are events 1, 5, and 9.

The complexes must be adjusted to fit events 4, 7, and 8 such that they are singly

```
1 2 3 5 6 9
```

covered This process proceeds as before for k=2 The steps taken during that process are not shown in this listing

```
RULE COVERING CLUSTER 1   (EVENTS 6 3 2 1)
RULE 216  EVENT SETS 1, COSTS(-6-2 3) -4 -41 0
 FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [MST-ONTOP(P1)] [ONTOP(P1,P2)] [SAMESIZE(P1 P2)]
[SHAPE(P1)#RECTANGLE] [SHAPE(P2)#SQUARE] [SIZE(P1)=1] [SIZE(P2)=1] [TEXTURE(P1)=*]
[TEXTURE(P2)=*] [#DIFF-SHAPE=3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1] [#PS(SHAPE=RECTANGLE)=0  1]
[#PS(SHAPE=DIAMOND)=0  1] [#PS(SHAPE=TRIANGLE)=0  1] [#PS(SHAPE=CIRCLE)#3]
[#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1] [#PS(SHAPE=POLYGON)=1  2]
[#PS(SIZE=1)=2] [#PS(SIZE=2)=0  1] [#PS(SIZE=0)#3] [#PS(TEXTURE=0)=2  3]
[#PS(TEXTURE=1)=1  2] [#PS=3  4]

RULE COVERING CLUSTER 2.  (EVENTS 5)
RULE 437  EVENT SETS 1, COSTS(-6-2 3) -1 -47 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [LST-ONTOP(P3)] [MST-ONTOP(P1)] [ONTOP(P1,P2)]
[ONTOP(P2,P3)] [SAMESIZE(P1 P3)] [SAMETEXTUR(P1 P3)] [SHAPE(P1)=CIRCLE]
[SHAPE(P2)=TRIANGLE] [SHAPE(P3)=USHAPE] [SIZE(P1)=2] [SIZE(P2)=1] [SIZE(P3)=2]
[TEXTURE(P2)=0] [TEXTURE(P3)=1] [TEXTURE(P1)=1] [#DIFF-SHAPE=3] [#DIFF-SIZE=2]
[#DIFF-TEXTURE=2] [#PS(SHAPE=ELLIPSE)=0] [#PS(SHAPE=USHAPE)=1]
[#PS(SHAPE=POLYGON)=1] [#PS(SHAPE=CURVED)=1] [#PS(SHAPE=SQUARE)=0]
[#PS(SHAPE=RECTANGLE)=0] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=1]
[#PS(SHAPE=CIRCLE)=1] [#PS(SIZE=1)=1] [#PS(SIZE=2)=2] [#PS(SIZE=0)=0]
[#PS(TEXTURE=0)=1] [#PS(TEXTURE=1)=2] [#PS=3]

RULE COVERING CLUSTER 3   (EVENTS 9 8 7 4)
RULE 1796  EVENT SETS 1, COSTS(-6-2 3) -4 -37 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)#CIRCLE] [SHAPE(P2)=CURVED] [SIZE(P1)=*]
[SIZE(P2)#0] [TEXTURE(P2)=1] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)=0  1] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SIZE=2)#3] [#PS(SIZE=0)=0  1] [#PS(SIZE=1)#3]
[#PS(TEXTURE=1)=2  3] [#PS(TEXTURE=0)=0  1] [#PS=2  3]

THE FOLLOWING EVENTS COULD NOT BE COVERED   (NONE)

FINDING NEWSEEDS
CNO=1 EVENT=1 DSUM=  4.2000000000000E+001
CNO=1 EVENT=2 DSUM=  4.2000000000000E+001
CNO=1 EVENT=3 DSUM=  3.6000000000000E+001
CNO=1 EVENT=6 DSUM=  4.0000000000000E+001
CNO=2 EVENT=5 DSUM=  0.0000000000000E+000
CNO=3 EVENT=4 DSUM=  5.4000000000000E+001
CNO=3 EVENT=4 DSUM=  5.4000000000000E+001
CNO=3 EVENT=7 DSUM=  6.2000000000000E+001
CNO=3 EVENT=8 DSUM=  6.0000000000000E+001
CNO=3 EVENT=9 DSUM=  7.8000000000000E+001
```

Here are the class descriptions of the three resulting clusters They are illustrated in the upper part of Fig 6 6

The sums of distances to other events are given in this table

```
ENTER NUMBER OF SEEDS FOLLOWED BY THEIR EVENT NUMBERS
? 3 3 5 4
NOW COVERING EVENT 3 AGAINST 5 4
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 3 2 1  (0 NEW)
RULE 1819  EVENT SETS 1, COSTS(-6-2 3) -4 -1 0
[#PS(TEXTURE=0)=2  3]

NOW COVERING EVENT 5 AGAINST 4 3
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 5  (0 NEW)
RULE 1839  EVENT SETS 1, COSTS(-6-2 3) -3 -1 0
[#PS(SIZE=1)=0  1]

NOW COVERING EVENT 4 AGAINST 5 3
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 8 7 4  (0 NEW)
RULE 1856  EVENT SETS 1, COSTS(-6-2 3) -3 -1 0
[#DIFF-SHAPE#3]

EVENTS COVERED
 1 2 3 6
EVENTS COVERED
 5 8 9
EVENTS COVERED
 4 7 8
MULTIPLY-COVERED EVENTS
 8
UNIQUELY DETERMINED
 1 2 3 4 5 6 7 9

RULE COVERING CLUSTER 1  (EVENTS 8 6 3 2 1)
RULE 2466  EVENT SETS 1, COSTS(-6-2 3) -5 -38 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)#CIRCLE] [SHAPE(P2)#SQUARE] [SIZE(P1)#0]
[SIZE(P2)=1] [TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=USHAPE)=0  1] [#PS(SHAPE=POLYGON)=1  2]
[#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1] [#PS(SHAPE=RECTANGLE)=0  1]
[#PS(SHAPE=DIAMOND)=0  1] [#PS(SHAPE=TRIANGLE)=0  1] [#PS(SHAPE=CIRCLE)#3]
[#PS(SHAPE=ELLIPSE)=0  1] [#PS(SIZE=2)=0  1] [#PS(SIZE=0)#3] [#PS(SIZE=1)=1  2]
[#PS(TEXTURE=1)=1  2] [#PS#0  1]

RULE COVERING CLUSTER 2  (EVENTS 9 5)
RULE 2211  EVENT SETS 1, COSTS(-6-2 3) -2 -44 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[LST-ONTOP(P3)] [MST-ONTOP(P1)] [ONTOP(P1,P2)] [ONTOP(P2,P3)] [SHAPE(P1)#RECTANGLE]
[SHAPE(P2)#SQUARE] [SHAPE(P3)=USHAPE] [SIZE(P1)=*] [SIZE(P2)#0] [SIZE(P3)=2]
[TEXTURE(P3)=1] [TEXTURE(P1)=1] [TEXTURE(P2)=*] [#DIFF-SHAPE=3] [#DIFF-SIZE=2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)=0  1] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=1]
[#PS(SHAPE=POLYGON)=1] [#PS(SHAPE=CURVED)=1] [#PS(SHAPE=SQUARE)=0  1]
[#PS(SHAPE=RECTANGLE)=0] [#PS(SIZE=1)=0  1] [#PS(SIZE=2)=2] [#PS(SIZE=0)=0  1]
[#PS(TEXTURE=0)=0  1] [#PS(TEXTURE=1)=2  3] [#PS=3]

RULE COVERING CLUSTER 3  (EVENTS 7 4)
RULE 2348  EVENT SETS 1, COSTS(-6-2 3) -2 -38 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
```

The next iteration is requested by giving the three seed event numbers corresponding to the events which have the minimum distances in the above table. They are events 9, 5, and 4.

The only multiply covered event is event 8. The complexes are adjusted to cover it uniquely. The steps of this process are not shown in this listing.

The following class descriptions give the second clustering. It is illustrated in the lower part of Fig. 6.6.

```
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[LST-ONTOP(P2)] [MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)#SQUARE]
[SHAPE(P2)=CURVED] [SIZE(P1)#0] [SIZE(P2)=1] [TEXTURE(P1)=*] [TEXTURE(P2)=1]
[#DIFF-SHAPE=2] [#DIFF-SIZE=1  2] [#DIFF-TEXTURE=1  2] [#PS(SHAPE=USHAPE)=0]
[#PS(SHAPE=POLYGON)=0  1] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0]
[#PS(SHAPE=RECTANGLE)=0] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SHAPE=ELLIPSE)=1] [#PS(SIZE=0)=0] [#PS(SIZE=1)=2]
[#PS(SIZE=2)=0  1] [#PS(TEXTURE=0)=0  1] [#PS(TEXTURE=1)=2] [#PS=2  3]


THE FOLLOWING EVENTS COULD NOT BE COVERED   (NONE)

FINDING NEW SEEDS
CNO=1 EVENT=1 DSUM=  5 70000000000000E+001
CNO=1 EVENT=2 DSUM=  5 80000000000000E+001
CNO=1 EVENT=3 DSUM=  5 30000000000000E-001
CNO=1 EVENT=6 DSUM=  5 60000000000000E-001
CNO=1 EVENT=8 DSUM=  7 00000000000000E-001
CNO=2 EVENT=5 DSUM=  1 70000000000000E-001
CNO=2 EVENT=9 DSUM=  1 70000000000000E-001
CNO=3 EVENT=4 DSUM=  1 70000000000000E+001
CNO=3 EVENT=4 DSUM=  1 70000000000000E-001
CNO=3 EVENT=7 DSUM=  3 40000000000000E+001

ENTER NUMBER OF SEEDS FOLLOWED BY THEIR EVENT NUMBERS
? 3 7 8 9
NOW COVERING EVENT 7 AGAINST 9 8
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 7 6 4 3 2 1  (0 NEW)
RULE 2737  EVENT SETS 1, COSTS(-6-2 3) -6 -1 0
[#PS(SIZE=1)=2  3]

NOW COVERING EVENT 8 AGAINST 9 7
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 8 5  (0 NEW)
RULE 2753  EVENT SETS 1, COSTS(-6-2 3) -2 -1 0
[#PS(SIZE=1)=1]

NOW COVERING EVENT 9 AGAINST 8 7
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 5  (0 NEW)
RULE 2771  EVENT SETS 1, COSTS(-6-2 3) -2 -1 0
[#PS(SIZE=2)=2  3]

EVENTS COVERED
 1 2 3 4 6 7
EVENTS COVERED
 5 8
EVENTS COVERED
 5 9
MULTIPLY-COVERED EVENTS
 5
UNIQUELY DETERMINED
 1 2 3 4 6 7 8 9

RULE COVERING CLUSTER 1   (EVENTS 8 7 6 5 4 3 2 1)
RULE 3384  EVENT SETS 1, COSTS(-6-2 3) -7 -36 1
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[MST-ONTOP(P1)] [ONTOP(P1,P2)] [SHAPE(P1)=*] [SHAPE(P2)#SQUARE] [SIZE(P1)#0]
[SIZE(P2)=1] [TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)#3] [#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1]
```

This table gives the sums of distances to the events

The next iteration is begun by entering the seed event numbers. The events 7, 8, and 9 are selected because they have the minimum distance values in the above table

Event 5 is multiply covered. Complexes will be adjusted to cover it uniquely

The following class descriptions present the generated clustering

```
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0  1] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)#3] [#PS(SIZE=2)#3] [#PS(SIZE=0)#3] [#PS(SIZE=1)=1  2]
[#PS(TEXTURE=1)=1  2] [#PS=0  1]

RULE COVERING CLUSTER 2   (EVENTS 8)
RULE 3033  EVENT SETS 1, COSTS(-8-2 3) -1 -42 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)] [LST-ONTOP(P2)] [MST-ONTOP(P1)] [ONTOP(P1,P2)]
[SAMETEXTUR(P1 P2)] [SHAPE(P1)=RECTANGLE] [SHAPE(P2)=CIRCLE] [SIZE(P1)=2]
[SIZE(P2)=1] [TEXTURE(P2)=1] [TEXTURE(P1)=1] [#DIFF-SHAPE=2] [#DIFF-SIZE=2]
[#DIFF-TEXTURE=1] [#PS(SHAPE=SQUARE)=0] [#PS(SHAPE=RECTANGLE)=1]
[#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0] [#PS(SHAPE=CIRCLE)=1]
[#PS(SHAPE=ELLIPSE)=0] [#PS(SHAPE=USHAPE)=0] [#PS(SHAPE=POLYGON)=1]
[#PS(SHAPE=CURVED)=1] [#PS(SIZE=2)=1] [#PS(SIZE=0)=0] [#PS(SIZE=1)=1]
[#PS(TEXTURE=1)=2] [#PS(TEXTURE=0)=0] [#PS=2]

RULE COVERING CLUSTER 3   (EVENTS 9)
RULE 3244  EVENT SETS 1, COSTS(-8-2 3) -1 -47 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)] [LST-ONTOP(P3)] [MST-ONTOP(P1)] [ONTOP(P1,P2)]
[ONTOP(P2,P3)] [SAMESIZE(P2 P3)] [SAMETEXTUR(P1 P2 P3)] [SHAPE(P1)=SQUARE]
[SHAPE(P2)=ELLIPSE] [SHAPE(P3)=USHAPE] [SIZE(P1)=0] [SIZE(P2)=2] [SIZE(P3)=2]
[TEXTURE(P2)=1] [TEXTURE(P3)=1] [TEXTURE(P1)=1] [#DIFF-SHAPE=3] [#DIFF-SIZE=2]
[#DIFF-TEXTURE=1] [#PS(SHAPE=ELLIPSE)=1] [#PS(SHAPE=USHAPE)=1]
[#PS(SHAPE=POLYGON)=1] [#PS(SHAPE=CURVED)=1] [#PS(SHAPE=SQUARE)=1]
[#PS(SHAPE=RECTANGLE)=0] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0]
[#PS(SHAPE=CIRCLE)=0] [#PS(SIZE=1)=0] [#PS(SIZE=2)=2] [#PS(SIZE=0)=1]
[#PS(TEXTURE=0)=0] [#PS(TEXTURE=1)=3] [#PS=3]

THE FOLLOWING EVENTS COULD NOT BE COVERED   (NONE)

FINDING NE.SEEDS
CNO=1 EVENT=1 DSUM=  1 03000000000000E-002
CNO=1 EVENT=2 DSUM=  1 10000000000000E-002
CNO=1 EVENT=3 DSUM=  9 60000000000000E-001
CNO=1 EVENT=4 DSUM=  1 04000000000000E-002
CNO=1 EVENT=4 DSUM=  1 04000000000000E-002
CNO=1 EVENT=5 DSUM=  1 20000000000000E-002
CNO=1 EVENT=6 DSUM=  1 02000000000000E-002
CNO=1 EVENT=7 DSUM=  1 11000000000000E-002
CNO=2 EVENT=8 DSUM=  0 00000000000000E-000
CNO=3 EVENT=9 DSUM=  0 00000000000000E-000

ENTER ONE CHAR. (P)ARAMETERS, (V)L1, (C)OVER, (M)ODIFY, (H)ELP FOR OPTIONS
? q
CPU TIME USED     173 169 SEC

TABLE OF PROCEDURE USE COUNTS
    614-ALLC            73-AQ             97-AQSET           9-ADDSEL
      0-ADDCONS        305-ABSOURB         0-APPLRULES       0-ADDLRULES
    314-ADDLINK       5628-COSTC           0-CLEANCPX        3-CLUSTER
     33-CCOVER        8748-COSTG          33-COVER           0-COSTL
      0-CALCARITH     1056-COMPMS       2580-EXTND        14214-FREETAB
   3776-FREEGRPH      6829-FREECPX        0-SUBGDETAIL      74-LQST2
      0-LSCHOOSE       33-ADDMETA        97-ADDTOMQ        688-NE.CPXTAB
   7075-NE.C          3786-NE.G         848-NE.GP           3-NE.SEEDS
   3872-COPYCPX       3872-COPYGRPH       0-PROCARITH     62096-SUBG1
  29085-SUBG           347-TRIMC        272-TRIMG           0-TRIML
     33-TRIMM          848-TOOSMALL        0-VL1           11-VLINT
```

The following table of sums of distances contains values greater than the last clustering. Since there is no improvement, the program is terminated

```
  58532-CPXCOV          283-GENRLIZ

                ARITH     SGWA    LRULE    CPXTB      CPX    GRAPH
 NEW COUNTS         0        1        0       10       70       35
NUMBER FREE         -        -        -       10       60       25
```

B.5. Results generated for k=4

```
INDUCE 3   VERSION AS OF AUGUST 25 1984

THIS RUN MADE ON  84 09 22
INITIALIZING   (MFIGS)

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
VARIABLE NAME   VTYPE   VCOST          TRACE
   #PT           LIN       1          STOPS
   #P            LIN       1          PRINT RULES  TRUE
   #DIFF         LIN       1
   SHAPE         STR       1
   SIZE          LIN       1

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PARAMETERS  GENERAL          VL              AQ            L-RULE
        REGENSTAR    1    VLMAXSTAR  2    AQMAXSTAR  4    MAXL     5
        METATRIM     1    NCONSIST   2    AQCUTF1   20
        DESCTYPE  DISC    ALTER      2    LQST   FALSE
        EXTMTY    TRUE    MINCOVER 100
        EQUIV     TRUE    MAXBACK    0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
           VLNF=3              AQNF=2              LRNF=4
CRITERION  VLCRIT    VLTOL     AQCRIT    AQTOL     LRCRIT    LRTOL
             -1       0 00       -1       0 00       -1       0 00
             -2       0 00        2       0 00        2       0 00
              3       0 00                           -3       0 00
                                                      4       0 00
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

ENTER ONE CHAR  (P)ARAMETERS, (V)LI, (C)OVER, (M)ODIFY, (H)ELP FOR OPTIONS
? u

ENTER NUMBER OF SEEDS FOLLOWED BY THEIR EVENT NUMBERS
? 4 1 2 3 4
NOW COVERING EVENT 1 AGAINST 4 3 2
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 8 5 1  (0 NEW)
RULE 22  EVENT SETS 1, COSTS(-6-2 3) -3 -2 0
[SIZE(P1)=2][#PS(SHAPE=CIRCLE)=1  3]

NOW COVERING EVENT 2 AGAINST 4 3 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 7 6 2  (0 NEW)
RULE 47  EVENT SETS 1, COSTS(-6-2 3) -3 -1 0
[#PS(SIZE=2)=0]

NOW COVERING EVENT 3 AGAINST 4 2 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 7 3  (0 NEW)
RULE 60  EVENT SETS 1, COSTS(-6-2 3) -3 -2 0
[#PS(SHAPE=ELLIPSE)#0][#PS(SHAPE=CURVED)#3]

NOW COVERING EVENT 4 AGAINST 3 2 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 8 7 4  (0 NEW)
```

Note that parameters VLMAXSTAR, NCONSIST and ALTER have been set to 2 to conserve cpu time

This response directs the program to construct four classes using the events numbered 1, 2, 3, and 4 as the initial seeds

These complexes results from the four stars built which cover each seed event against the other seed events

```
RULE 86  EVENT SETS 1, COSTS(-6-2 3) -3 -1 0
[#DIFF-SHAPE#3.

EVENTS COVERED
 1 5 8
EVENTS COVERED
 2 6 7
EVENTS COVERED
 3 7 9
EVENTS COVERED
 4 7 8
MULTIPLY-COVERED EVENTS
 7 8
UNIQUELY DETERMINED
 1 2 3 4 5 6 9

RULE COVERING CLUSTER 1   (EVENTS 8 7 5 1)
RULE 546  EVENT SETS 1, COSTS(-6-2 3) -4 -38 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE] [MST-ONTOP(P1)]
[ONTOP(P1,P2)] [SHAPE(P1)#ELLIPSE] [SHAPE(P2)#SQUARE] [SIZE(P1)#0] [SIZE(P2)=1]
[TEXTURE(P2)=*] [TEXTURE(P1)=*] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)=1] [#PS(SHAPE=CURVED)=1] [#PS(SHAPE=SQUARE)=0]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0  1] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)=0  1] [#PS(SIZE=2)#3] [#PS(SIZE=0)=0] [#PS(SIZE=1)=1  2]
[#PS(TEXTURE=1)=1  2] [#PS(TEXTURE=0)#3] [#PS=2  3]

RULE COVERING CLUSTER 2   (EVENTS 6 2)
RULE 208  EVENT SETS 1, COSTS(-6-2 3) -2 -41 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [MST-ONTOP(P1)] [ONTOP(P1,P2)] [SAMESIZE(P1 P2)]
[SHAPE(P1)#RECTANGLE] [SHAPE(P2)=RECTANGLE] [SIZE(P1)=1] [SIZE(P2)=1]
[TEXTURE(P1)=*] [TEXTURE(P2)=0] [#DIFF-SHAPE=3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=CURVED)=2  3] [#PS(SHAPE=SQUARE)=0  1] [#PS(SHAPE=RECTANGLE)=1]
[#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0] [#PS(SHAPE=CIRCLE)=2]
[#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0] [#PS(SHAPE=POLYGON)=1  2]
[#PS(SIZE=1)=2] [#PS(SIZE=2)=0] [#PS(SIZE=0)=2] [#PS(TEXTURE=0)=2  3]
[#PS(TEXTURE=1)=1  2] [#PS=4]

RULE COVERING CLUSTER 3   (EVENTS 9 3)
RULE 269  EVENT SETS 1, COSTS(-6-2 3) -2 -44 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[LST-ONTOP(P3)] [MST-ONTOP(P1)] [ONTOP(P1,P2)] [ONTOP(P2,P3)] [SHAPE(P1)=POLYGON]
[SHAPE(P2)#SQUARE] [SHAPE(P3)#POLYGON] [SIZE(P1)#2] [SIZE(P2)#0] [SIZE(P3)=2]
[TEXTURE(P3)=*] [TEXTURE(P1)=*] [TEXTURE(P2)=1] [#DIFF-SHAPE=3] [#DIFF-SIZE=2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)=0] [#PS(SHAPE=ELLIPSE)=1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)=1  2] [#PS(SHAPE=CURVED)=1] [#PS(SHAPE=SQUARE)=0  1]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SIZE=1)#3] [#PS(SIZE=2)=1  2] [#PS(SIZE=0)=0  1]
[#PS(TEXTURE=0)#3] [#PS(TEXTURE=1)#0] [#PS=3]

RULE COVERING CLUSTER 4   (EVENTS 4)
RULE 346  EVENT SETS 1, COSTS(-6-2 3) -1 -49 0
 FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
```

The complexes must be adjusted by NID to uniquely fit events 7 and 8 to complexes

The following complexes are the result of applying NID

```
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)] [FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=1)=FALSE]
[FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [LST-ONTOP(P2)] [LST-ONTOP(P3)] [MST-ONTOP(P1)]
[ONTOP(P1,P3)] [ONTOP(P1,P2)] [SAMESHAPE(P2 P3)] [SAMESIZE(P2 P3)]
[SAMETEXTUR(P2 P3)] [SHAPE(P1)=ELLIPSE] [SHAPE(P2)=CIRCLE] [SHAPE(P3)=CIRCLE]
[SIZE(P1)=2] [SIZE(P2)=1] [SIZE(P3)=1] [TEXTURE(P1)=0] [TEXTURE(P2)=1]
[TEXTURE(P3)=1] [#DIFF-SHAPE=2] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=POLYGON)=0] [#PS(SHAPE=CURVED)=3] [#PS(SHAPE=SQUARE)=0]
[#PS(SHAPE=RECTANGLE)=0] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0]
[#PS(SHAPE=CIRCLE)=2] [#PS(SHAPE=ELLIPSE)=1] [#PS(SHAPE=USHAPE)=0] [#PS(SIZE=0)=0]
[#PS(SIZE=1)=2] [#PS(SIZE=2)=1] [#PS(TEXTURE=0)=1] [#PS(TEXTURE=1)=2] [#PS=3]

THE FOLLOWING EVENTS COULD NOT BE COVERED   (NONE)

FINDING NEW SEEDS

CNO=1 EVENT=1 DSUM=  4 70000000000000E+001
CNO=1 EVENT=5 DSUM=  4 50000000000000E+001
CNO=1 EVENT=7 DSUM=  4 80000000000000E+001
CNO=1 EVENT=8 DSUM=  4 40000000000000E+001
CNO=2 EVENT=2 DSUM=  1 10000000000000E+001
CNO=2 EVENT=6 DSUM=  1 10000000000000E+001
CNO=3 EVENT=3 DSUM=  2 10000000000000E+001
CNO=3 EVENT=9 DSUM=  2 10000000000000E+001
CNO=4 EVENT=4 DSUM=  0 00000000000000E+000
CNO=4 EVENT=4 DSUM=  0 00000000000000E+000

ENTER NUMBER OF SEEDS FOLLOWED BY THEIR EVENT NUMBERS
? 4 5 6 9 4
NOW COVERING EVENT 5 AGAINST 9 6 4
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 8 5  (0 NEW)
RULE 689  EVENT SETS 1, COSTS(-6-2 3) -2 -1 0
[#PS(SIZE=1)=1]

NOW COVERING EVENT 6 AGAINST 9 5 4
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 6 3 2 1  (0 NEW)
RULE 708  EVENT SETS 1, COSTS(-6-2 3) -4 -1 0
[#PS(TEXTURE=0)=2  3]

NOW COVERING EVENT 9 AGAINST 6 5 4
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 9 8 7  (0 NEW)
RULE 731  EVENT SETS 1, COSTS(-6-2 3) -3 -1 0
[#PS(TEXTURE=0)=0]

NOW COVERING EVENT 4 AGAINST 9 6 5
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 8 7 4  (0 NEW)
RULE 753  EVENT SETS 1, COSTS(-6-2 3) -3 -1 0
[#DIFF-SHAPE#3]

EVENTS COVERED
 5 8
EVENTS COVERED
 1 2 3 6
EVENTS COVERED
 7 8 9
EVENTS COVERED
 4 7 8
MULTIPLY-COVERED EVENTS
 7 8
UNIQUELY DETERMINED
 1 2 3 4 5 6 9
```

The sums of distances to other events are given in this table.

For the second iteration, the seed with the lowest distance sum is selected from each cluster developed in the previous iteration. The selected seeds are events 5, 6, 9, and 4.

The following complexes are the result of applying NID to fit the multiply covered events (7 and 8) to the complexes These class descriptions are illustrated in Fig 6 7

```
RULE COVERING CLUSTER 1   (EVENTS 8 7 5)
RULE 1218  EVENT SETS 1, COSTS(-6-2 3) -3 -38 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE] [MST-ONTOP(P1)]
[ONTOP(P1,P2)] [SHAPE(P1)#ELLIPSE] [SHAPE(P2)#SQUARE] [SIZE(P1)#0] [SIZE(P2)=1]
[TEXTURE(P2)=*] [TEXTURE(P1)=1] [#DIFF-SHAPE=2  3] [#DIFF-SIZE=1  2]
[#DIFF-TEXTURE=1  2] [#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1]
[#PS(SHAPE=POLYGON)=1] [#PS(SHAPE=CURVED)=1] [#PS(SHAPE=SQUARE)=0]
[#PS(SHAPE=RECTANGLE)=0  1] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0  1]
[#PS(SHAPE=CIRCLE)=0  1] [#PS(SIZE=2)#3] [#PS(SIZE=0)=0] [#PS(SIZE=1)=1  2]
[#PS(TEXTURE=1)=2] [#PS(TEXTURE=0)=0  1] [#PS=2  3]

RULE COVERING CLUSTER 2   (EVENTS 6 3 2 1)
RULE 874  EVENT SETS 1, COSTS(-6-2 3) -4 -41 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [MST-ONTOP(P1)] [ONTOP(P1,P2)] [SAMESIZE(P1 P2)]
[SHAPE(P1)#RECTANGLE] [SHAPE(P2)#SQUARE] [SIZE(P1)=1] [SIZE(P2)=1] [TEXTURE(P1)=*]
[TEXTURE(P2)=*] [#DIFF-SHAPE=3] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=CURVED)#0] [#PS(SHAPE=SQUARE)=0  1] [#PS(SHAPE=RECTANGLE)=0  1]
[#PS(SHAPE=DIAMOND)=0  1] [#PS(SHAPE=TRIANGLE)=0  1] [#PS(SHAPE=CIRCLE)#3]
[#PS(SHAPE=ELLIPSE)=0  1] [#PS(SHAPE=USHAPE)=0  1] [#PS(SHAPE=POLYGON)=1  2]
[#PS(SIZE=1)=2] [#PS(SIZE=2)=0  1] [#PS(SIZE=0)#3] [#PS(TEXTURE=0)=2  3]
[#PS(TEXTURE=1)=1  2] [#PS=3  4]

RULE COVERING CLUSTER 3   (EVENTS 9)
RULE 942  EVENT SETS 1, COSTS(-6-2 3) -1 -47 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)=FALSE] [FORALL-PS(SIZE=0)=FALSE]
[FORALL-PS(SIZE=1)=FALSE] [FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)] [LST-ONTOP(P3)] [MST-ONTOP(P1)] [ONTOP(P1,P2)]
[ONTOP(P2,P3)] [SAMESIZE(P2 P3)] [SAMETEXTUR(P1 P2 P3)] [SHAPE(P1)=SQUARE]
[SHAPE(P2)=ELLIPSE] [SHAPE(P3)=USHAPE] [SIZE(P1)=0] [SIZE(P2)=2] [SIZE(P3)=2]
[TEXTURE(P2)=1] [TEXTURE(P3)=1] [TEXTURE(P1)=1] [#DIFF-SHAPE=3] [#DIFF-SIZE=2]
[#DIFF-TEXTURE=1] [#PS(SHAPE=ELLIPSE)=1] [#PS(SHAPE=USHAPE)=1]
[#PS(SHAPE=POLYGON)=1] [#PS(SHAPE=CURVED)=1] [#PS(SHAPE=SQUARE)=1]
[#PS(SHAPE=RECTANGLE)=0] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0]
[#PS(SHAPE=CIRCLE)=0] [#PS(SIZE=1)=0] [#PS(SIZE=2)=2] [#PS(SIZE=0)=1]
[#PS(TEXTURE=0)=0] [#PS(TEXTURE=1)=3] [#PS=3]

RULE COVERING CLUSTER 4:  (EVENTS 4)
RULE 1016:  EVENT SETS 1, COSTS(-6-2 3) -1 -49 0
[FORALL-PS(SHAPE=SQUARE)=FALSE] [FORALL-PS(SHAPE=RECTANGLE)=FALSE]
[FORALL-PS(SHAPE=DIAMOND)=FALSE] [FORALL-PS(SHAPE=TRIANGLE)=FALSE]
[FORALL-PS(SHAPE=CIRCLE)=FALSE] [FORALL-PS(SHAPE=ELLIPSE)=FALSE]
[FORALL-PS(SHAPE=USHAPE)=FALSE] [FORALL-PS(SHAPE=POLYGON)=FALSE]
[FORALL-PS(SHAPE=CURVED)] [FORALL-PS(SIZE=0)=FALSE] [FORALL-PS(SIZE=1)=FALSE]
[FORALL-PS(SIZE=2)=FALSE] [FORALL-PS(TEXTURE=0)=FALSE]
[FORALL-PS(TEXTURE=1)=FALSE] [LST-ONTOP(P2)] [LST-ONTOP(P3)] [MST-ONTOP(P1)]
[ONTOP(P1,P3)] [ONTOP(P1,P2)] [SAMESHAPE(P2 P3)] [SAMESIZE(P2 P3)]
[SAMETEXTUR(P2,P3)] [SHAPE(P1)=ELLIPSE] [SHAPE(P2)=CIRCLE] [SHAPE(P3)=CIRCLE]
[SIZE(P1)=2] [SIZE(P2)=1] [SIZE(P3)=1] [TEXTURE(P1)=0] [TEXTURE(P2)=1]
[TEXTURE(P3)=1] [#DIFF-SHAPE=2] [#DIFF-SIZE=2] [#DIFF-TEXTURE=2]
[#PS(SHAPE=POLYGON)=0] [#PS(SHAPE=CURVED)=3] [#PS(SHAPE=SQUARE)=0]
[#PS(SHAPE=RECTANGLE)=0] [#PS(SHAPE=DIAMOND)=0] [#PS(SHAPE=TRIANGLE)=0]
[#PS(SHAPE=CIRCLE)=2] [#PS(SHAPE=ELLIPSE)=1] [#PS(SHAPE=USHAPE)=0] [#PS(SIZE=0)=0]
[#PS(SIZE=1)=2] [#PS(SIZE=2)=1] [#PS(TEXTURE=0)=1] [#PS(TEXTURE=1)=2] [#PS=3]
```

```
THE FOLLOWING EVENTS COULD NOT BE COVERED   (NONE)

DOING NEWSEEDS

CNO=1 EVENT=5 DSUM=  3.3000000000000E+001
CNO=1 EVENT=7 DSUM=  3.0000000000000E+001
CNO=1 EVENT=8 DSUM=  2.7000000000000E+001
CNO=2 EVENT=1 DSUM=  4.2000000000000E+001
CNO=2 EVENT=2 DSUM=  4.2000000000000E+001
CNO=2 EVENT=3 DSUM=  3.6000000000000E+001
CNO=2 EVENT=6 DSUM=  4.0000000000000E+001
CNO=3 EVENT=9 DSUM=  0.0000000000000E+000
CNO=4 EVENT=4 DSUM=  0.0000000000000E+000
CNO=4 EVENT=4 DSUM=  0.0000000000000E-000

ENTER ONE CHAR  (P)ARAMETERS, (V)L1, (C)OVER, (M)ODIFY, (H)ELP FOR OPTIONS   The quit directive is given
? q
CPU TIME USED     74.131 SEC

TABLE OF PROCEDURE USE COUNTS
    814-ALLC            69-AQ             93-AQSET           9-ADDSEL
      0-ADDCONS        317-ABSOURB         0-APPLRULES       0-ADDLRULES
    314-ADDLINK       6935-COSTC           0-CLEANCPX        2-CLUSTER
     32-CCOVER        3538-COSTG          32-COVER           0-COSTL
      0-CALCARITH     1024-COMPMS       3163-EXTND       11060-FREETAB
   1427-FREEGRPH      5144-FREECPX        0-SUBGDETAIL      60-LQST2
      0-LSCHOOSE       32-ADDMETA        93-ADDTOMQ       874-NEWCPXTAB
   5505-NEWC          1437-NEWG         461-NEWGP          2-NEWSEEDS
   1515-COPYCPX       1515-COPYGRPH       0-PROCARITH    14642-SUBG1
  10463-SUBG           477-TRIMC        285-TRIMG          0-TRIML
     32-TRIMM          461-TOOSMALL       0-VL1           11-VLINT
  69981-CPXCOV         204-GENRLIZ

                ARITH     SGWA    LRULE    CPXTB     CPX    GRAPH
 NEW COUNTS         0        1        0       10      72       41
NUMBER FREE         -        -        -       10      62       31
```

APPENDIX C

CLASSIFYING TOY TRAINS

C.1. Problem description

The toy trains pictured in Fig. 7.3 are described by the following predicates and functions.

CCONT(T,C) — train T contains car C

CSHAPE(C) — the shape of car C, one of

OPENRECT	open rectangle
OPENTRAP	open trapezoid
USHAPE	U-shape
DBLOPNRECT	double open rectangle
OPENTOP	any one of the above shapes
HEXAGON	hexagon
ELLIPSE	ellipse
CLOSEDRECT	closed rectangle
JAGGEDTOP	jagged top rectangle
SLOPETOP	sloping top rectangle
ENGINE	locomotive
CLOSEDTOP	any of the above shapes

INFRONT(C1,C2) — car C1 is in front of car C2

LN(C) — the length of car C, one of

SHORT	the car is short
LONG	the car is long

LSHAPE(C) — the shape of the cargo in car C, one of

RECTANGLOD	rectangular load
TRIANGLOD	triangular load
HEXAGONLOD	hexagonal load
CIRCLELOD	circular load

NCAR(T) — the number of cars in train T

NPL(C)	the number of cargo items in car C
NWHL(C)	number of wheels on each side of car C
WTYPE	the type of wheel on car C, one of

BL	all wheels black
CL	all wheels clear
MXD	a mixture of black and clear wheels

C.2. Input data file

```
TRAINS
M
A
[NCAR(T1)=4..4][NWHL(CAR1)=2..2][LN(CAR1)=SHORT..LONG]
[NPL(CAR1)=1..1][CSHAPE(CAR1)=OPENRECT,OPENTRAP,USHAPE,DBLOPNRECT,
HEXAGON,ELLIPSE,CLOSEDRECT,JAGGEDTOP,SLOPETOP,ENGINE]=>[D=1].
M
D
Y
E
[CSHAPE=OPENRECT,OPENTRAP,USHAPE,DBLOPNRECT]=>[CSHAPE=OPENTOP].
E
[CSHAPE=HEXAGON,ELLIPSE,CLOSEDRECT,JAGGEDTOP,SLOPETOP,ENGINE]
    =>[CSHAPE=CLOSEDTOP].
```

These 8 lines are used to establish the domains of the functions NCAR, NWHL, LN, NPL, and CSHAPE. The rule entered with M and A is deleted with M, D, and Y

The E directive introduces each value inference rule. The first rule defines OPENTOP as a generalization of the values OPENRECT, OPENTRAP, USHAPE, and DBLOPNRECT. The second rule defines the generalized value CLOSEDTOP

```
M
A
[CCONT(T1,CAR1)][CCONT(T1,CAR2)][CCONT(T1,CAR3)]
[CCONT(T1,CAR4)][CCONT(T1,CAR5)]
[NCAR(T1)=5]
[INFRONT(CAR1,CAR2)][INFRONT(CAR2,CAR3)][INFRONT(CAR3,CAR4)]
[INFRONT(CAR4,CAR5)][NWHL(CAR1)=2][NWHL(CAR2)=2][NWHL(CAR3)=2]
[NWHL(CAR4)=3][NWHL(CAR5)=2][WTYPE(CAR1)=BK][WTYPE(CAR2)=BK]
[WTYPE(CAR3)=BK][WTYPE(CAR4)=BK][WTYPE(CAR5)=BK][LN(CAR1)=LONG]
[LN(CAR2)=LONG][LN(CAR3)=SHORT][LN(CAR4)=LONG][LN(CAR5)=SHORT]
[CSHAPE(CAR1)=ENGINE][CSHAPE(CAR2)=OPENRECT][CSHAPE(CAR3)=SLOPETOP]
[CSHAPE(CAR4)=OPENRECT][CSHAPE(CAR5)=OPENRECT]
[NPL(CAR1)=0][NPL(CAR2)=3][NPL(CAR3)=1][NPL(CAR4)=1][NPL(CAR5)=1]
[LSHAPE(CAR2)=RECTANGLOD][LSHAPE(CAR3)=TRIANGLOD]
[LSHAPE(CAR4)=HEXAGONLOD][LSHAPE(CAR5)=CIRCLELOD]
=>[D=1].
```

Each train event is described by a complex introduced with the M and A directives. This description is for train A in Fig. 7.9

```
M
A
[CCONT(T1,CAR1)][CCONT(T1,CAR2)][CCONT(T1,CAR3)][CCONT(T1,CAR4)]
[NCAR(T1)=4]
```

This description is for train B.

```
[INFRONT(CAR1,CAR2)][INFRONT(CAR2,CAR3)][INFRONT(CAR3,CAR4)]
[NWHL(CAR1)=2][NWHL(CAR2)=2][NWHL(CAR3)=2][NWHL(CAR4)=2]
[WTYPE(CAR1)=BK][WTYPE(CAR2)=CL][WTYPE(CAR3)=CL][WTYPE(CAR4)=CL]
[LN(CAR1)=LONG][LN(CAR2)=SHORT][LN(CAR3)=SHORT][LN(CAR4)=SHORT]
[CSHAPE(CAR1)=ENGINE][CSHAPE(CAR2)=USHAPE][CSHAPE(CAR3)=OPENTRAP]
[CSHAPE(CAR4)=CLOSEDRECT]
[NPL(CAR1)=0][NPL(CAR2)=1][NPL(CAR3)=1][NPL(CAR4)=2]
[LSHAPE(CAR2)=TRIANGLOD][LSHAPE(CAR3)=RECTANGLOD][LSHAPE(CAR4)=CIRCLELOD]
=>[D=1].
M                                                    This description is for train C
A
[CCONT(T1,CAR1)][CCONT(T1,CAR2)][CCONT(T1,CAR3)][CCONT(T1,CAR4)]
[NCAR(T1)=4]
[INFRONT(CAR1,CAR2)][INFRONT(CAR2,CAR3)][INFRONT(CAR3,CAR4)]
[NWHL(CAR1)=2][NWHL(CAR2)=2][NWHL(CAR3)=2][NWHL(CAR4)=3]
[WTYPE(CAR1)=BK][WTYPE(CAR2)=CL][WTYPE(CAR3)=CL][WTYPE(CAR4)=CL]
[LN(CAR1)=LONG][LN(CAR2)=SHORT][LN(CAR3)=SHORT][LN(CAR4)=LONG]
[CSHAPE(CAR1)=ENGINE][CSHAPE(CAR2)=OPENRECT][CSHAPE(CAR3)=HEXAGON]
[CSHAPE(CAR4)=CLOSEDRECT]
[NPL(CAR1)=0][NPL(CAR2)=1][NPL(CAR3)=1][NPL(CAR4)=1]
[LSHAPE(CAR2)=CIRCLELOD][LSHAPE(CAR3)=TRIANGLOD][LSHAPE(CAR4)=TRIANGLOD]
=>[D=1].
M                                                    This description is for train D
A
[CCONT(T1,CAR1)][CCONT(T1,CAR2)][CCONT(T1,CAR3)][CCONT(T1,CAR4)][CCONT(T1,CAR5)]
[NCAR(T1)=5]
[INFRONT(CAR1,CAR2)][INFRONT(CAR2,CAR3)][INFRONT(CAR3,CAR4)][INFRONT(CAR4,CAR5)]
[NWHL(CAR1)=2][NWHL(CAR2)=2][NWHL(CAR3)=2][NWHL(CAR4)=2][NWHL(CAR5)=2]
[WTYPE(CAR1)=CL][WTYPE(CAR2)=CL][WTYPE(CAR3)=CL][WTYPE(CAR4)=CL][WTYPE(CAR5)=CL]
[LN(CAR1)=LONG][LN(CAR2)=SHORT][LN(CAR3)=SHORT][LN(CAR4)=SHORT][LN(CAR5)=SHORT]
[CSHAPE(CAR1)=ENGINE][CSHAPE(CAR2)=OPENTRAP][CSHAPE(CAR3)=DBLOPNRECT]
[CSHAPE(CAR4)=ELLIPSE][CSHAPE(CAR5)=OPENRECT]
[NPL(CAR1)=0][NPL(CAR2)=1][NPL(CAR3)=1][NPL(CAR4)=1][NPL(CAR5)=1]
[LSHAPE(CAR2)=TRIANGLOD][LSHAPE(CAR3)=TRIANGLOD][LSHAPE(CAR4)=RECTANGLOD]
[LSHAPE(CAR5)=RECTANGLOD]
=>[D=1].
M                                                    This description is for train E
A
[CCONT(T1,CAR1)][CCONT(T1,CAR2)][CCONT(T1,CAR3)][CCONT(T1,CAR4)]
[NCAR(T1)=4]
[INFRONT(CAR1,CAR2)][INFRONT(CAR2,CAR3)][INFRONT(CAR3,CAR4)]
[NWHL(CAR1)=2][NWHL(CAR2)=2][NWHL(CAR3)=3][NWHL(CAR4)=2]
[WTYPE(CAR1)=MXD][WTYPE(CAR2)=BK][WTYPE(CAR3)=BK][WTYPE(CAR4)=BK]
[LN(CAR1)=LONG][LN(CAR2)=SHORT][LN(CAR3)=LONG][LN(CAR4)=SHORT]
[CSHAPE(CAR1)=ENGINE][CSHAPE(CAR2)=DBLOPNRECT][CSHAPE(CAR3)=CLOSEDRECT]
[CSHAPE(CAR4)=CLOSEDRECT]
[NPL(CAR1)=0][NPL(CAR2)=1][NPL(CAR3)=1][NPL(CAR4)=1]
[LSHAPE(CAR2)=TRIANGLOD][LSHAPE(CAR3)=RECTANGLOD][LSHAPE(CAR4)=CIRCLELOD]
=>[D=1].
M                                                    This description is for train F
A
[CCONT(T1,CAR1)][CCONT(T1,CAR2)][CCONT(T1,CAR3)]
```

```
[NCAR(T1)=3]
[INFRONT(CAR1,CAR2)][INFRONT(CAR2,CAR3)]
[NWHL(CAR1)=2][NWHL(CAR2)=2][NWHL(CAR3)=2]
[WTYPE(CAR1)=BK][WTYPE(CAR2)=CL][WTYPE(CAR3)=CL]
[LN(CAR1)=LONG][LN(CAR2)=LONG][LN(CAR3)=SHORT]
[CSHAPE(CAR1)=ENGINE][CSHAPE(CAR2)=CLOSEDRECT][CSHAPE(CAR3)=OPENRECT]
[NPL(CAR1)=0][NPL(CAR2)=3][NPL(CAR3)=1]
[LSHAPE(CAR2)=CIRCLELOD][LSHAPE(CAR3)=TRIANGLOD]
=>[D=1].
M                                                    This description is for train
A                                                    G
[CCONT(T1,CAR1)][CCONT(T1,CAR2)][CCONT(T1,CAR3)][CCONT(T1,CAR4)]
[NCAR(T1)=4]
[INFRONT(CAR1,CAR2)][INFRONT(CAR2,CAR3)][INFRONT(CAR3,CAR4)]
[NWHL(CAR1)=2][NWHL(CAR2)=2][NWHL(CAR3)=2][NWHL(CAR4)=2]
[WTYPE(CAR1)=CL][WTYPE(CAR2)=BK][WTYPE(CAR3)=CL][WTYPE(CAR4)=CL]
[LN(CAR1)=LONG][LN(CAR2)=SHORT][LN(CAR3)=SHORT][LN(CAR4)=LONG]
[CSHAPE(CAR1)=ENGINE][CSHAPE(CAR2)=DBLOPNRECT][CSHAPE(CAR3)=USHAPE]
[CSHAPE(CAR4)=JAGGEDTOP]
[NPL(CAR1)=0][NPL(CAR2)=1][NPL(CAR3)=1][NPL(CAR4)=0]
[LSHAPE(CAR2)=CIRCLELOD][LSHAPE(CAR3)=TRIANGLOD]
=>[D=1].
M                                                    This description is for train H
A
[CCONT(T1,CAR1)][CCONT(T1,CAR2)][CCONT(T1,CAR3)]
[NCAR(T1)=3]
[INFRONT(CAR1,CAR2)][INFRONT(CAR2,CAR3)]
[NWHL(CAR1)=2][NWHL(CAR2)=3][NWHL(CAR3)=2]
[WTYPE(CAR1)=CL][WTYPE(CAR2)=CL][WTYPE(CAR3)=CL]
[LN(CAR1)=LONG][LN(CAR2)=LONG][LN(CAR3)=SHORT]
[CSHAPE(CAR1)=ENGINE][CSHAPE(CAR2)=CLOSEDRECT][CSHAPE(CAR3)=USHAPE]
[NPL(CAR1)=0][NPL(CAR2)=1][NPL(CAR3)=1]
[LSHAPE(CAR2)=RECTANGLOD][LSHAPE(CAR3)=CIRCLELOD]
=>[D=1].
M                                                    This description is for train I
A
[CCONT(T1,CAR1)][CCONT(T1,CAR2)][CCONT(T1,CAR3)][CCONT(T1,CAR4)][CCONT(T1,CAR5)]
[NCAR(T1)=5]
[INFRONT(CAR1,CAR2)][INFRONT(CAR2,CAR3)][INFRONT(CAR3,CAR4)][INFRONT(CAR4,CAR5)]
[NWHL(CAR1)=2][NWHL(CAR2)=2][NWHL(CAR3)=2][NWHL(CAR4)=2][NWHL(CAR5)=2]
[WTYPE(CAR1)=CL][WTYPE(CAR2)=CL][WTYPE(CAR3)=CL][WTYPE(CAR4)=CL][WTYPE(CAR5)=CL]
[LN(CAR1)=LONG][LN(CAR2)=SHORT][LN(CAR3)=LONG][LN(CAR4)=SHORT][LN(CAR5)=SHORT]
[CSHAPE(CAR1)=ENGINE][CSHAPE(CAR2)=OPENTRAP][CSHAPE(CAR3)=JAGGEDTOP]
[CSHAPE(CAR4)=OPENRECT][CSHAPE(CAR5)=OPENTRAP]
[NPL(CAR1)=0][NPL(CAR2)=1][NPL(CAR3)=1][NPL(CAR4)=1][NPL(CAR5)=1]
[LSHAPE(CAR2)=CIRCLELOD][LSHAPE(CAR3)=RECTANGLOD][LSHAPE(CAR4)=RECTANGLOD]
[LSHAPE(CAR5)=CIRCLELOD]
=>[D=1].
M                                                    This description is for train J
A
[CCONT(T1,CAR1)][CCONT(T1,CAR2)][CCONT(T1,CAR3)]
[NCAR(T1)=3]
```

```
[INFRONT(CAR1,CAR2)][INFRONT(CAR2,CAR3)]
[NWHL(CAR1)=2][NWHL(CAR2)=2][NWHL(CAR3)=2]
[WTYPE(CAR1)=CL][WTYPE(CAR2)=CL][WTYPE(CAR3)=CL]
[LN(CAR1)=LONG][LN(CAR2)=SHORT][LN(CAR3)=LONG]
[CSHAPE(CAR1)=ENGINE][CSHAPE(CAR2)=USHAPE][CSHAPE(CAR3)=OPENRECT]
[NPL(CAR1)=0][NPL(CAR2)=1][NPL(CAR3)=2]
[LSHAPE(CAR2)=RECTANGLOD][LSHAPE(CAR3)=RECTANGLOD]
=>[D=1].
P
EXTMTY
EQUIV
METATRIM 99
ALTER 5
VLMAXSTAR 4
VLNF 2
VLCRIT(1) -1
VLCRIT(2) -2
VLTOL(1) 0
VLTOL(2) 0
DESCTYPE CHAR
PA
QUIT
```

These parameter settings enable the construction of selectors by application of counting quantified variables, counting similar selectors, counting distinct values, universal properties, chain properties, and equivalence properties transformations (see Sec 4.2). The last two transformations are enabled by the EXTMTY and EQUIV directives, respectively. The other transformations are enabled by the METATRIM directive. The DESCTYPE CHAR directive causes the program to build a structural template characterization.

C.3. Results from structural template generation

```
INDUCE 3 - VERSION AS OF AUGUST 25 1984

THIS RUN MADE ON  84/09/18
INITIALIZING  (MTRAINS)

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
VARIABLE NAME   VTYPE   VCOST        TRACE: 3 10
   #PT           LIN       1        STOPS:
   #P            LIN       1        PRINT RULES: TRUE
   #DIFF         LIN       1
   NCAR          LIN       1
   NWHL          LIN       1
   LN            LIN       1
   NPL           LIN       1
   CSHAPE        STR       1

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PARAMETERS: GENERAL          VL              AQ           L-RULE
       REGENSTAR    1    VLMAXSTAR  4    AQMAXSTAR  2    MAXL     5
       METATRIM    99    NCONSIST   4    AQCUTF1   20
       DESCTYPE  CHAR    ALTER      5    LQST   FALSE
       EXTMTY    TRUE    MINCOVER 100
       EQUIV     TRUE    MAXBACK    0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
          VLNF=3            AQNF=2            LRNF=4
CRITERION: VLCRIT    VLTOL    AQCRIT    AQTOL    LRCRIT    LRTOL
```

This report gives the names of functions which have a domain which is not nominal. The VTYPE code LIN denotes a linear domain, the code STR denotes a structured domain with a value hierarchy.

This report gives the values of control parameters.

This report shows the evaluation criterion LEF components. The most important

```
    3        30%      -1        0 00      -1        0 00
   -1        0 00      2        0 00       2        0 00
    2        0 00                         -3        0 00
                                           4        0 00
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

LEF is the one at the VL level, under the column with the VLCRIT heading. Criterion -1 maximizes the number of events covered. Criterion -2 maximizes the number of of selectors in the complex (to request that the complex be maximally specific).

```
ENTER ONE CHAR: (P)ARAMETERS, (V)L1, (C)OVER, (M)ODIFY, (H)ELP FOR OPTIONS
? c

WHAT EVENT SET DO YOU WANT TO COVER?
ENTER NUMBER OR SYMBOLIC VALUE OF DECISION VARIABLE
? 1

NOW COVERING EVENT SET 1
THE FOLLOWING RULES COVER SET 1
THIS RULE COVERS EVENTS 13 12 11 10 9 8 7 6 5 4  (10 NEW)
RULE 489  EVENT SETS 1, COSTS( 3-1 2) 0 -10 84
[CCONT(T1,CAR1)][CCONT(T1,CAR2)][CCONT(T1,CAR3)][CSHAPE(CAR1)=ENGINE]
[CSHAPE(CAR2)#HEXAGON][CSHAPE(CAR3)#ELLIPSE][FORALL-CARS(NPL=3)=FALSE]
[FORALL-CARS(CSHAPE=OPENRECT)=FALSE][FORALL-CARS(CSHAPE=OPENTRAP)=FALSE]
[FORALL-CARS(CSHAPE=USHAPED)=FALSE][FORALL-CARS(CSHAPE=DBLOPNRECT)=FALSE]
[FORALL-CARS(CSHAPE=HEXAGON)=FALSE][FORALL-CARS(CSHAPE=ELLIPSE)=FALSE]
[FORALL-CARS(CSHAPE=CLOSEDRECT)=FALSE][FORALL-CARS(CSHAPE=JAGGEDTOP)=FALSE]
[FORALL-CARS(CSHAPE=SLOPETOP)=FALSE][FORALL-CARS(CSHAPE=ENGINE)=FALSE]
[FORALL-CARS(CSHAPE=OPENTOP)=FALSE][FORALL-CARS(CSHAPE=CLOSEDTOP)=FALSE]
[FORALL-CARS(WTYPE=MXD)=FALSE][FORALL-CARS(LSHAPE=TRIANGLOD)=FALSE]
[FORALL-CARS(LSHAPE=HEXAGONLOD)=FALSE][FORALL-CARS(LSHAPE=CIRCLELOD)=FALSE]
[FORALL-CARS(NWHL=3)=FALSE][FORALL-CARS(LN=SHORT)=FALSE]
[FORALL-CARS(LN=LONG)=FALSE][FORALL-CARS(NPL=0)=FALSE]
[FORALL-CARS(NPL=1)=FALSE][FORALL-CARS(NPL=2)=FALSE][INFRONT(CAR2,CAR3)]
[INFRONT(CAR1,CAR2)][LN(CAR1)=LONG][LN(CAR2)=*][LN(CAR3)=*]
[LSHAPE(CAR2)#HEXAGONLOD][LSHAPE(CAR3)#HEXAGONLOD][MST-INFRON(CAR1),
[NCAR(T1)=3  5][NPL(CAR1)=0][NPL(CAR2)#0][NPL(CAR3)=1  2][NWHL(CAR1)=2]
[NWHL(CAR2)=2  3][NWHL(CAR3)=2  3][WTYPE(CAR1)=*][WTYPE(CAR2)#MXD]
[WTYPE(CAR3)#MXD][#CARS(CSHAPE=HEXAGON)=0  1][#CARS(CSHAPE=ELLIPSE)=0  1]
[#CARS(CSHAPE=CLOSEDRECT)=0  2][#CARS(CSHAPE=JAGGEDTOP)=0  1]
[#CARS(CSHAPE=SLOPETOP)=0  1][#CARS(CSHAPE=ENGINE)=1]
[#CARS(CSHAPE=OPENTOP)=1  3][#CARS(CSHAPE=CLOSEDTOP)=1  3]
[#CARS(CSHAPE=OPENRECT)#4  5][#CARS(CSHAPE=OPENTRAP)=0  2]
[#CARS(CSHAPE=USHAPED)=0  1][#CARS(CSHAPE=DBLOPNRECT)=0  1][#CARS(LN=SHORT)#0,5]
[#CARS(LN=LONG)=1  3][#CARS(LSHAPE=RECTANGLOD)=0  2]
[#CARS(LSHAPE=TRIANGLOD)=0  2][#CARS(LSHAPE=HEXAGONLOD)=0  1]
[#CARS(LSHAPE=CIRCLELOD)=0  2][#CARS(NPL=0)=1  2][#CARS(NPL=1)#0,5]
[#CARS(NPL=2)=0  1][#CARS(NPL=3)=0  1][#CARS(NWHL=2)#0  1][#CARS(NWHL=3)=0  1]
[#CARS(WTYPE=MXD)=0  1][#CARS=3  5][#DIFF-CSHAPE=3  5][#DIFF-LN=2]
[#DIFF-LSHAPE#0,5][#DIFF-NCAR=1][#DIFF-NPL=2  3][#DIFF-NWHL=1  2]
[#DIFF-WTYPE=1  2][#TS(NCAR=5)=0  1][#TS(NCAR=3)=0  1][#TS(NCAR=4)=0  1][#TS=1]
```

The c response requests a characteristic cover of the events.

All events are input as members of class 1.

All 10 trains are covered by this characteristic description. This structural template binds some of the quantified variables to certain parts of objects. Predicates and functions which take different values in at least two events are used to build a derived attribute-based description, shown below in Sec. C.4.

```
ENTER ONE CHAR: (P)ARAMETERS, (V)L1, (C)OVER, (M)ODIFY, (H)ELP FOR OPTIONS
? q

CPU TIME USED:     89.574 SEC

TABLE OF PROCEDURE USE COUNTS:
     10-ALLC              0-AQ              1-AQSET          10-ADDSEL
      0-ADDCONS          26-ABSOURB         1-APPLRULES       0-ADDLRULES
    958-ADDLINK           0-COSTC           0-CLEANCPX        0-CLUSTER
      0-CCOVER         1323-COSTG           1-COVER           0-COSTL
      0-CALCARITH        69-COMPMS          0-EXTND        1013-FREETAB
    488-FREEGRPH        488-FREECPX         0-SUBGDETAIL      10-LQST2
      1-LSCHOOSE          1-ADDMETA         1-ADDTOMQ         20-NEWCPXTAB
    507-NEWC            499-NEWG           93-NEWGP           0-NEWSEEDS
    486-COPYCPX         486-COPYGRPH        0-PROCARITH     14753-SUBG1
   7866-SUBG              0-TRIMC          25-TRIMG           0-TRIML
```

The above template description is all that is needed from INDUCE/9.

About 82 seconds of CPU time on a Cyber 175 was needed to build the structural template.

1-TRIMM 93-TOOSMALL 0-VL1 13-VLINT
0-CPXCOV 66-GENRLIZ

	ARITH	SGWA	LRULE	CPXTB	CPX	GRAPH
NEW COUNTS	0	1	0	11	34	34
NUMBER FREE	-	-	-	11	23	23

C.4. Attributes derived from template matching

The following tables of 51 attribute values for each of the 10 trains were derived by matching the above structural characterization template to the events. In the tables, attributes derived from predicates take the values 0 and 1 to denote FALSE and TRUE, respectively. Here, event numbers 1 to 10 correspond to trains A to J, respectively.

EVENTS

#	CSHAPE2	CSHAPE3	FANWHL2	FAWTYPEBK	FAWTYPECL	FALSHAPERECT	LN2	LN3	LSHAPE2
1	OPENRECT	SLOPETOP	0	1	0	0	LONG	SHORT	RECTLOD
2	USHAPE	OPENTRAP	1	0	0	0	SHORT	SHORT	TRILOD
3	OPENRECT	HEXAGON	0	0	0	0	SHORT	SHORT	CIRLOD
4	OPENTRAP	DBLOPRECT	1	0	1	0	SHORT	SHORT	TRILOD
5	DBLOPRECT	CLOSEDRECT	0	0	0	0	SHORT	LONG	TRILOD
6	CLOSEDRECT	OPENRECT	1	0	0	0	LONG	SHORT	CIRLOD
7	DBLOPRECT	USHAPE	1	0	0	0	SHORT	SHORT	CIRLOD
8	CLOSEDRECT	USHAPE	0	0	1	0	LONG	SHORT	RECTLOD
9	OPENTRAP	JAGGEDTOP	1	0	1	0	SHORT	LONG	CIRLOD
10	USHAPE	OPENRECT	1	0	1	1	SHORT	LONG	RECTLOD

The attribute names are abbreviated. The leading FA denotes "forall". It is followed by the property, written without punctuation or equals sign, e.g., FANWHL2 denotes "forall cars number of wheels equals 2". On attribute names not beginning with FA, an ending numeral denotes the car number, e.g., 2 denotes car2 as in CSHAPE2.

EVENTS

#	LSHAPE3	NCAR	NPL2	NPL3	NWHL2	SAMENWHL123	SAMEWTYPE123	WTYPE3	WTYPE1	WTYPE2
1	TRILOD	5	3	1	2	1	1	BK	BK	BK
2	RECTLOD	4	1	1	2	1	0	CL	BK	CL
3	TRILOD	4	1	1	2	1	0	CL	BK	CL
4	TRILOD	5	1	1	2	1	1	CL	CL	CL
5	RECTLOD	4	1	1	2	0	0	BK	MXD	BK
6	TRILOD	3	3	1	2	1	0	CL	BK	CL
7	TRILOD	4	1	1	2	1	0	CL	CL	BK
8	CIRLOD	3	1	1	3	0	1	CL	CL	CL
9	RECTLOD	5	1	1	2	1	1	CL	CL	CL
10	RECTLOD	3	1	2	2	1	1	CL	CL	CL

The abbreviation beginning with SAME denotes a "same" predicate. Ending numerals denote the arguments, e.g., SAMENWHL123 denotes "same number of wheels for car1, car2, and car3". NCAR stands for "number of cars in the train".

EVENTS

#	NCARSCLOSDREC	NCARSJAGGED	NCARSSLOPE	NCARSOPENTOP	NCARSCLOSDTOP
1	0	0	1	3	2
2	1	0	0	2	2
3	1	0	0	1	3
4	0	0	0	3	2
5	2	0	0	1	3
6	1	0	0	1	2
7	0	1	0	2	2
8	1	0	0	1	2
9	0	1	0	3	2
10	0	0	0	2	1

The abbreviation beginning NCARS denotes the number of cars with the particular property coded in the remainder of the attribute name. These attributes here give the numbers of cars with particular car shapes.

EVENTS

#	NCARSOPENREC	NCARSOPENTRAP	NCARSUSHAPE	NCARSDBLOPREC	NCARSHEXAGON
1	3	0	0	0	0
2	0	1	1	0	0
3	1	0	0	0	1
4	1	1	0	1	0
5	0	0	0	1	0
6	1	0	0	0	0
7	0	0	1	1	0
8	0	0	1	0	0
9	1	2	0	0	0

This table continues giving attributes involving the number of cars of a particular shape.

10	1	0	1	0	0	

EVENTS

#	NCARSELLIPSE	NCARSSHORT	NCARSLONG	NCARSHEXLOD	NCARSCIRLOD	NCARSRECTLOD
1	0	2	3	1	1	1
2	0	3	1	0	1	1
3	0	2	2	0	1	0
4	1	4	1	0	0	2
5	0	2	2	0	1	1
6	0	1	2	0	1	0
7	0	2	2	0	1	0
8	0	1	2	0	1	1
9	0	3	2	0	2	2
10	0	1	2	0	0	2

The attributes NCARSSHORT and NCARSLONG give the number of short and long cars, respectively. The attributes ending in LOD give the number of cars carrying various shapes of cargo (loads).

EVENTS

#	NCARSTRILOD	NCARSNPL2	NCARSNPL3	NCARSNPL0	NCARSNPL1	NCARSNWHL2	NCARSNWHL3
1	1	0	1	1	3	4	1
2	1	1	0	1	2	4	0
3	2	3	1	1	3	3	1
4	2	0	0	1	4	5	0
5	1	0	1	1	3	3	1
6	1	0	1	1	1	3	0
7	1	0	0	2	2	4	0
8	0	0	0	1	2	2	1
9	0	0	0	1	4	5	0
10	0	1	0	1	1	3	0

The attributes ending NPL0, NPL1, NPL2, NPL3 give the number of cars carrying 0, 1, 2, and 3 units of cargo, respectively. The attributes ending NWHL2 and NWHL3 give the number of cars with 2 and 3 wheels (on one side), respectively.

EVENTS

#	NCARSWTYPEMXD	NCARSWTYPECL	NCARSWTYPEBK	DIFFCSHAPE	DIFFLSHAPE	DIFFNPL
1	0	0	5	3	4	3
2	0	3	1	4	3	3
3	0	3	1	4	2	2
4	0	5	0	5	2	2
5	1	0	3	3	3	2
6	0	2	1	3	2	3
7	0	3	1	4	2	2
8	0	3	0	3	2	2
9	0	5	0	4	2	2
10	0	3	0	3	1	3

The attributes involving WTYPEMXD, WTYPECL, and WTYPEBK give the number of cars that have mixed, all clear, and all black wheels, respectively. The abbreviation DIFF is used on attributes that give the number of different values for a function used to describe a particular train.

EVENTS

#	DIFFNWHL	DIFFWTYPE	LSTINFRONT3
1	2	1	0
2	1	2	0
3	2	2	0
4	1	1	0
5	2	2	0
6	1	2	1
7	1	2	0
8	2	1	1
9	1	1	0
10	1	1	1

The attribute LSTINFRONT3 is derived from the LEASTinfront predicate (LSTINFRON) evaluated for car3. In trains 6, 8, and 10, car3 is the end car represented by a true (1) value for this attribute.

C.5. Attributes added to describe structural fragments

As described in Sec. 7.3, seven attributes are added to describe structural units in the trains not covered by the structural template. The first attribute gives the number of one-car additions attached to car3. This attribute is called '#1at3'. The second attribute gives the number of two-car additions attached to car3. It is called '#2at3'. The remaining 5 attributes are conditional attributes which describe the part attached to car3 (which is called car4 for convenience). The value UNK (unknowable) is used when a train does not contain any part taking the role of car4.

EVENTS

#	#1at3	#2at3	CSHAPE4	LSHAPE4	NPL4	N#HL4	#TYPE4
1	0	1	OPENRECT	HEXLOD	1	3	BK
2	1	0	CLOSDRECT	CIRLOD	2	2	CL
3	1	0	CLOSDRECT	TRILOD	1	3	CL
4	0	1	ELLIPSE	RECTLOD	1	2	CL
5	1	0	CLOSDRECT	CIRLOD	1	2	BK
6	0	0	UNK	UNK	UNK	UNK	UNK
7	1	0	JAGGEDTOP	UNK	0	2	CL
8	0	0	UNK	UNK	UNK	UNK	UNK
9	0	1	OPENRECT	RECTLOD	1	2	CL
10	0	0	UNK	UNK	UNK	UNK	UNK

C.6. Input file for attribute-based clustering

The following specifications are input data for the attribute-based clustering program CLUSTER/2. The input file consists of the following declarative statements (in the form of relational tables) plus the attribute-based event descriptions, as shown above. Since the above tables of attribute values are shown in the format used by CLUSTER/2, they are not duplicated in the listing of the input file. A comment at the end of the input file listing shows where the event descriptions (in table form) are normally placed.

```
#MAINTITLE  "CLUSTERING TRAINS USING 58 DERIVED ATTRIBUTES"

PARAMETERS
TRACE  CRITERION MIDSPEED COVERTYPE    BASE PROBE H1 H2 H3 K
 OFF        DS      SLOW   DISJOINT      2    20  4  4  4 2
```

These parameters request a clustering into two classes. The PROBE value is set large (20) to extend the search for good clusterings to improve the results.

```
  DS-CRITERION
#  CRITERION TOLERANCE
1   -COM         0 95
2    DIM         0 0
```

The clustering criterion LEF is to minimize the number of selectors (negative commonality) and then maximize the dimensionality (number of discriminant variables). The tolerance on the first criterion is set very large at 95%. It is used to screen out exceptionally specific clusterings (e.g., a singleton cluster and a 9-event cluster) which have high dimensionality scores but are really not very interesting.

VARIABLES

#	NAME	TYPE	LEVELS
1	CSHAPE2	STR	5
2	CSHAPE3	STR	8
3	FAN#HL2	NOM	2
4	FA#TYPEBK	NOM	2
5	FA#TYPECL	NOM	2

The VARIABLES (attributes) table gives the type and number of values in the domain of each attribute. NOM denotes a nominal domain, LIN denotes a linear domain, STR denotes a struc-

tured domain with a value hierarchy

6	FALSHAPERECT	NOM	2
7	LN2	NOM	2
8	LN3	NOM	2
9	LSHAPE2	NOM	3
10	LSHAPE3	NOM	3
11	NCAR	LIN	3
12	NPL2	NOM	2
13	NPL3	NOM	2
14	NWHL2	NOM	2
15	SAMENWHL123	NOM	2
16	SAMEWTYPE123	NOM	2
17	WTYPE3	NOM	2
18	WTYPE1	NOM	3
19	WTYPE2	NOM	2
20	NCARSCLOSDREC	LIN	3
21	NCARSJAGGED	NOM	2
22	NCARSSLOPE	NOM	2
23	NCARSOPENTOP	LIN	3
24	NCARSCLOSDTOP	LIN	3
25	NCARSOPENREC	LIN	4
26	NCARSOPENTRAP	LIN	3
27	NCARSUSHAPE	NOM	2
28	NCARSDBLOPREC	NOM	2
29	NCARSHEXAGON	NOM	2
30	NCARSELLIPSE	NOM	2
31	NCARSSHORT	LIN	4
32	NCARSLONG	LIN	3
33	NCARSHEXLOD	NOM	2
34	NCARSCIRLOD	LIN	3
35	NCARSRECTLOD	LIN	3
36	NCARSTRILOD	LIN	3
37	NCARSNPL2	LIN	4
38	NCARSNPL3	NOM	2
39	NCARSNPLO	NOM	2
40	NCARSNPL1	LIN	4
41	NCARSNWHL2	LIN	4
42	NCARSNWHL3	NOM	2
43	NCARSWTYPEMXD	NOM	2
44	NCARSWTYPECL	LIN	6
45	NCARSWTYPEBK	LIN	6
46	DIFFCSHAPE	LIN	3
47	DIFFLSHAPE	LIN	4
48	DIFFNPL	NOM	2
49	DIFFNWHL	NOM	2
50	DIFFWTYPE	NOM	2
51	LSTINFRONT3	NOM	2
52	#1at3	NOM	2
53	#2at3	NOM	2
54	CSHAPE4	STR	5
55	LSHAPE4	STR	5
56	NPL4	STR	4

57	N_HL4	STR	3
58	_TYPE4	STR	3

CSHAPE2-NAMES

VALUE	NAME
0	OPENRECT
1	USHAPE
2	OPENTRAP
3	DBLOPRECT
4	CLOSEDRECT

CSHAPE2-STRUCTURE

NAME	SUBNAME	SUBNAME	SUBNAME	SUBNAME
OPENTOP	OPENRECT	USHAPE	OPENTRAP	DBLOPRECT
CLOSTOP	CLOSEDRECT	$	$	$

The following tables called "-NAMES" are used to map the values in a domain with d values to the integers 0 to d-1. The tables called "-STRUCTURE" define the value hierarchy for a structured domain. In such tables, the values under the SUBNAME headings may be generalized to the value under the NAME heading. The $ is a null entry used as a place holder.

CSHAPE3-NAMES

VALUE	NAME
0	SLOPETOP
1	OPENTRAP
2	HEXAGON
3	DBLOPRECT
4	CLOSEDRECT
5	OPENRECT
6	USHAPE
7	JAGGEDTOP

CSHAPE3-STRUCTURE

NAME	SUBNAME	SUBNAME	SUBNAME	SUBNAME
OPENTOP	OPENTRAP	DBLOPRECT	OPENRECT	USHAPE
CLOSTOP	SLOPETOP	HEXAGON	CLOSEDRECT	JAGGEDTOP

LN2-NAMES

VALUE	NAME
0	SHORT
1	LONG

LN3-NAMES

VALUE	NAME
0	SHORT
1	LONG

LSHAPE2-NAMES

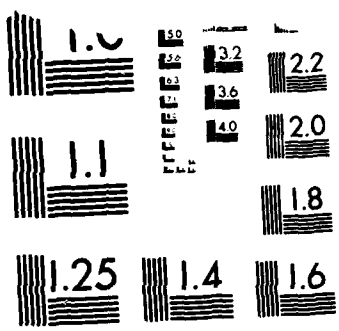
VALUE	NAME
0	RECTLOD
1	TRILOD
2	CIRLOD

LSHAPE3-NAMES

VALUE	NAME
0	TRILOD
1	RECTLOD

AD-A185 747 CONJUNCTIVE CONCEPTUAL CLUSTERING: A METHODOLOGY AND EXPERIMENTATION(U) ILLINOIS UNIV AT URBANA COORDINATED SCIENCE LAB R E STEPP SEP 87 UILU-ENG-87-2253 1/3

UNCLASSIFIED N00014-82-K-0186 F/G 12/9 NL



2	CIRLOD

NCAR-NAMES

VALUE	NAME
0	3
1	4
2	5

NPL2-NAMES

VALUE	NAME
0	1
1	3

NPL3-NAMES

VALUE	NAME
0	1
1	2

NWHL2-NAMES

VALUE	NAME
0	2
1	3

WTYPE3-NAMES

VALUE	NAME
0	BK
1	CL

WTYPE1-NAMES

VALUE	NAME
0	BK
1	CL
2	MXD

WTYPE2-NAMES

VALUE	NAME
0	BK
1	CL

NCARSOPENTOP-NAMES

VALUE	NAME
0	1
1	2
2	3

NCARSCLOSDTOP-NAMES

VALUE	NAME
0	1
1	2
2	3

NCARSSHORT-NAMES

VALUE NAME
0 1
1 2
2 3
3 4

NCARSLONG-NAMES
VALUE NAME
0 1
1 2
2 3

NCARSNPLO-NAMES
VALUE NAME
0 1
1 2

NCARSNPL1-NAMES
VALUE NAME
0 1
1 2
2 3
3 4

NCARSNWHL2-NAMES
VALUE NAME
0 2
1 3
2 4
3 5

DIFFCSHAPE-NAMES
VALUE NAME
0 3
1 4
2 5

DIFFLSHAPE-NAMES
VALUE NAME
0 1
1 2
2 3
3 4

DIFFNPL-NAMES
VALUE NAME
0 2
1 3

DIFFNWHL-NAMES
VALUE NAME
0 1

```
  1    2

DIFFWTYPE-NAMES
VALUE NAME
  0    1
  1    2

CSHAPE4-NAMES
VALUE NAME
  0    UNK
  1    OPENRECT
  2    CLOSDRECT
  3    ELLIPSE
  4    JAGGEDTOP

LSHAPE4-NAMES
VALUE NAME
  0    UNK
  1    HEXLOD
  2    CIRLOD
  3    TRILOD
  4    RECTLOD

NPL4-NAMES
VALUE NAME
  0    UNK
  1    0
  2    1
  3    2

NWHL4-NAMES
VALUE NAME
  0    UNK
  1    2
  2    3

WTYPE4-NAMES
VALUE NAME
  0    UNK
  1    CL
  2    BK

CSHAPE4-STRUCTURE
NAME      SUBNAME  SUBNAME  SUBNAME  SUBNAME
KNOWABL  OPENRECT CLOSDRECT ELLIPSE JAGGEDTOP
CLOSDTOP CLOSDRECT ELLIPSE JAGGEDTOP    $

LSHAPE4-STRUCTURE
NAME      SUBNAME  SUBNAME  SUBNAME  SUBNAME
KNOWABL   HEXLOD   CIRLOD   TRILOD   RECTLOD

NPL4-STRUCTURE
```

```
NAME      SUBNAME  SUBNAME  SUBNAME
KNOWABL     0        1        2

NWHL4-STRUCTURE
NAME      SUBNAME  SUBNAME
KNOWABL     2        3

WTYPE4-STRUCTURE
NAME      SUBNAME  SUBNAME
KNOWABL     CL       BK
```

Note: the EVENT tables shown in Sec. C.4 are inserted here in the input file.

C.7. Results from attribute-based clustering of trains

The following output is generated from the input file by the program CLUSTER/2.

```
CONJUNCTIVE CONCEPTUAL CLUSTERING PROGRAM CLUSTER/2   LAST UPGRADE: 04/10/84
  PARAMETERS
   MINK MAXK   TRACE   H1   H2   H3  INITMETHOD  WIDSPEED  COVERTYPE  CRITERION  BASE  PROBE  BETA
     2    2     ON      4    4    4    RANDOM      SLOW     DISJOINT   DS          2     20    3.0

  DS-CRITERION
    #   CRITERION  TOLERANCE
    1   -COM        0.95
    2    DIM        0.00

          CLUSTERING TRAINS USING 58 DERIVED ATTRIBUTES

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

 EXPERIMENT 1: K=2, CRITERION=DS


 THE 2 BEST CLUSTERINGS FOLLOW...    (266239 MS)

 ITER/CPLX#  VL-RULE                                    SEED -------COSTS--------
                                                              -COM       DIM
   9     1   [SAMEWTYPE123=1][WTYPE1=BK,CL]              4       12        -2   This clustering is based on the
             [NCARSCLOSDREC=0  1]                                               number of different wheel
             [NCARSCLOSDTOP=1  2][NCARSHEXAGON=0]                               types in the train. It is illus-
             [NCARSRECTLOD=1  2][NCARSNPL2=0  1]                                trated in Fig. 7.6B. Cluster 1
             [NCARSNPLO=1][NCARSWTYPEMXD=0]                                     contains trains with only one
             [DIFFWTYPE=1][NEXTRA1CAR=0][NPL4<>0]                               type of wheel.
             EVENTS COVERED: 1,4,8,9,10
   9     2   [CSHAPE2<>OPENTRAP]                         2       33        -2   Cluster 2 contains trains with
             [CSHAPE3<>SLOPETOP][FAWTYPEBK=0]                                   two types of wheels.
             [FAWTYPECL=0][FALSHAPERECT=0]
             [LSHAPE2=TRILOD,CIRLOD]
             [LSHAPE3=TRILOD,RECTLOD][NCAR=3  4]
             [NPL3=1][NWHL2=2][SAMEWTYPE123=0]
```

[NCARSSLOPE=0] [NCARSOPENTOP=1..2]
[NCARSCLOSDTOP=2..3]
[NCARSOPENREC=0..1]
[NCARSOPENTRAP=0..1] [NCARSELLIPSE=0]
[NCARSSHORT=1..3] [NCARSLONG=1..2]
[NCARSHEXLOD=0] [NCARSCIRLOD=1]
[NCARSRECTLOD=0..1] [NCARSTRILOD=1..2]
[NCARSNPL1=1..3] [NCARSNWHL2=3..4]
[NCARSWTYPECL=0..3]
[NCARSWTYPEBK=1..3] [DIFFCSHAPE=3..4]
[DIFFLSHAPE=2..3] [DIFFWTYPE=2]
[NEXTRA2CAR=0] [CSHAPE4<>OPENRECT]
[LSHAPE4<>HEXLOD]
EVENTS COVERED: 2,3,5,6,7

9	TOTALS			45	-4	
28	1	[CSHAPE2=OPENTOP] [CSHAPE3<>SLOPETOP] [FAWTYPEBK=0] [FALSHAPERECT=0] [LN2=SHORT] [LSHAPE2=TRILOD,CIRLOD] [LSHAPE3=TRILOD,RECTLOD] [NCAR=4..5] [NPL2=1] [NPL3=1] [NWHL2=2] [SAMENWHL123=1] [WTYPE3=CL] [WTYPE1=BK,CL] [NCARSCLOSDREC=0..1] [NCARSSLOPE=0] [NCARSCLOSDTOP=2..3] [NCARSOPENREC=0..1] [NCARSSHORT=2..4] [NCARSLONG=1..2] [NCARSHEXLOD=0] [NCARSNPL1=2..4] [NCARSNWHL2=3..5] [NCARSWTYPEWXD=0] [NCARSWTYPECL=3..5] [NCARSWTYPEBK=0..1] [DIFFCSHAPE=4..5] [DIFFLSHAPE=2..3] [LSTINFRONT3=0] [CSHAPE4=KNOWABL] [NPL4=KNOWABL] [NWHL4=KNOWABL] [WTYPE4=CL] EVENTS COVERED: 2,3,4,7,9	2	33	-2	*The alternative clustering is based on the number of different car shapes in the train. It is illustrated in Fig. 7.6A. Cluster 1 contains trains with 4 to 5 different car shapes and each train has a fourth car with clear wheels.*
28	2	[NCARSJAGGED=0] [NCARSOPENTRAP=0] [NCARSHEXAGON=0] [NCARSELLIPSE=0] [NCARSSHORT=1..2] [NCARSLONG=2..3] [NCARSCIRLOD=0..1] [NCARSTRILOD=0..1] [NCARSNPL2=0..1] [NCARSNPLO=1] [NCARSNPL1=1..3] [NCARSNWHL2=2..4] [NCARSWTYPECL=0..3] [DIFFCSHAPE=3] [NPL4<>0] [WTYPE4<>CL] EVENTS COVERED: 1,5,6,8,10	5	16	-2	*Cluster 2 contains trains with 3 different car shapes and the fourth car does not have clear wheels (this includes cases where the fourth car does not exist)*
28	TOTALS			49	-4	

(208 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S= 8.4E+022

ACTIVITY STATISTICS

CENDIST	=	36	CRITVAL	=	18578	CLUSTER	=	35			
CMPCOV	=	0	EXTEND	=	6582	FREECPLX	=	8598	GENRLIZ	=	1556

```
INTRSCT   =  5215  MID       =   517  NUMSEL    = 18578  NEWCPLX   =  8872
REDUCE    =  6582  REFHIGH   =115512  REFNUM    =     0  REFUNION  = 38167
REFLEV    =1371740 REFLOW    =161291  SETMAP     = 18578  SETLEVELMA=     0
STAR      =   208  SEMANTICS =  1449  SYNDIST   =     0  TRIM      =   208
TCOVER    =251600  DEGISCT   =     0  CLUSTERING=     1  TABLESETUP=    49
BESTC     =  3614  MCARD     = 30539  GENPATH    =   519  CLEARCV   =     2
SAVECV    =    35  STAR2     =     0
 268.564 CP SECS, 154035B CM USED.
```

C.8. Results from attribute-based clustering of trains with toxic predicate

The following output is generated from the input file after the addition of the TOXIC chemical descriptor to each train and the adjustment of descriptor costs to give the TOXIC attribute a highly favorable cost.

```
CONJUNCTIVE CONCEPTUAL CLUSTERING PROGRAM CLUSTER/2   LAST UPGRADE: 04/10/84



PARAMETERS
 MINK MAXK   TRACE   H1   H2   H3  INITMETHOD  NIDSPEED  COVERTYPE  CRITERION  BASE  PROBE  BETA  MAXHEIGHT  MINSIZE
   2    4     ON      4    4    4    RANDOM      SLOW    HIERARCHIAL DS          2     2    3.0      99         4



DS-CRITERION
  #   CRITERION  TOLERANCE
  1    SIM         0.90
  2    DIM         0.00


                TRAINS DESCRIBED BY 59 DESCRIPTORS
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1: K=2, CRITERION=DS

THE 1 BEST CLUSTERINGS FOLLOW      (26329 MS)

ITER CPLX#  VL-RULE                                          SEED -------COSTS--------
                                                                       SIM       DIM
 1    1   [CSHAPE3<>OPENTRAP][FALSHAPERECT=0]                  1        -90       -1     This clustering is illustrated in
          [LSHAPE2=RECTLOD,CIRLOD][NPL3=1]                                              Fig. 7.7.
          [WTYPE1=BK,CL][NCARSCLOSDREC=0..1]
          [NCARSCLOSDTOP=2..3][NCARSELLIPSE=0]
          [NCARSSHORT=1..3][NCARSLONG=2..3]
          [NCARSCIRLOD=1..2][NCARSNPL1=2..4]
          [NCARSWTYPEMXD=0][DIFFCSHAPE=3..4]
          [DIFFLSHAPE=2..4][TOXIC=T]
          EVENTS COVERED: 1,3,7,8,9
```

```
 1     2  [CSHAPE3<>SLOPETOP][FAWTYPEBK=0]          2          -91         -1
          [LSHAPE3=TRILOD,RECTLOD][NWHL2=2]
          [NCARSJAGGED=0][NCARSSLOPE=0]
          [NCARSOPENREC=0..1]
          [NCARSOPENTRAP=0..1][NCARSHEXAGON=0]
          [NCARSLONG=1..2][NCARSHEXLOD=0]
          [NCARSCIRLOD=0..1][NCARSNPL2=0..1]
          [NCARSNPLO=1][NCARSNWHL2=3..5]
          [NCARSWTYPEBK=0..3][DIFFLSHAPE=1..3]
          [CSHAPE4<>OPENRECT][NWHL4<>3]
          [TOXIC=F]
          EVENTS COVERED: 2,4,5,6,10
 1    TOTALS                                                -181         -2

(16 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  1.0E+023

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1: K=3, CRITERION=DS

THE 2 BEST CLUSTERINGS FOLLOW ...   (104880 MS)

ITER/CPLX#  VL-RULE                                    SEED -------COSTS--------
                                                              SIM         DIM
 1     1  [CSHAPE2=OPENRECT][CSHAPE3=SLOPETOP]          1          -100          0
          [FAWWHL2=0][FAWTYPEBK=1][FAWTYPECL=0]
          [FALSHAPERECT=0][LN2=LONG][LN3=SHORT]
          [LSHAPE2=RECTLOD][LSHAPE3=TRILOD]
          [NCAR=5][NPL2=3][NPL3=1][NWHL2=2]
          [SAMENWHL123=1][SAMEWTYPE123=1]
          [WTYPE3=BK][WTYPE1=BK][WTYPE2=BK]
          [NCARSCLOSDREC=0][NCARSJAGGED=0]
          [NCARSSLOPE=1][NCARSOPENTOP=3]
          [NCARSCLOSDTOP=2][NCARSOPENREC=3]
          [NCARSOPENTRAP=0][NCARSUSHAPE=0]
          [NCARSDBLOPREC=0][NCARSHEXAGON=0]
          [NCARSELLIPSE=0][NCARSSHORT=2]
          [NCARSLONG=3][NCARSHEXLOD=1]
          [NCARSCIRLOD=1][NCARSRECTLOD=1]
          [NCARSTRILOD=1][NCARSNPL2=0]
          [NCARSNPL3=1][NCARSNPLO=1]
          [NCARSNPL1=3][NCARSNWHL2=4]
          [NCARSNWHL3=1][NCARSWTYPEMXD=0]
          [NCARSWTYPECL=0][NCARSWTYPEBK=5]
          [DIFFCSHAPE=3][DIFFLSHAPE=4]
          [DIFFNPL=3][DIFFNWHL=2][DIFFWTYPE=1]
          [LSTINFRONT3=0][NEXTRA1CAR=0]
          [NEXTRA2CAR=1][CSHAPE4=OPENRECT]
          [LSHAPE4=HEXLOD][NPL4=1][NWHL4=3]
          [WTYPE4=BK][TOXIC=T]
```

		EVENTS COVERED: 1			
1	2	[CSHAPE3<>SLOPETOP] [FAWTYPEBK=0] [LSHAPE3=TRILOD,RECTLOD] [NWHL2=2] [NCARSJAGGED=0] [NCARSSLOPE=0] [NCARSOPENREC=0..1] [NCARSOPENTRAP=0..1] [NCARSHEXAGON=0] [NCARSLONG=1..2] [NCARSHEXLOD=0] [NCARSCIRLOD=0..1] [NCARSNPL2=0..1] [NCARSNPLO=1] [NCARSNWHL2=3..5] [NCARSWTYPEBK=0..3] [DIFFLSHAPE=1..3] [CSHAPE4<>OPENRECT] [NWHL4<>3] [TOXIC=F]	2	-91	0
		EVENTS COVERED: 2,4,5,6,10			
1	3	[CSHAPE3<>SLOPETOP] [FAWTYPEBK=0] [FALSHAPERECT=0] [LSHAPE2=RECTLOD, CIRLOD] [NPL2=1] [NPL3=1] [WTYPE3=CL] [WTYPE1=BK,CL] [NCARSCLOSDREC=0..1] [NCARSSLOPE=0] [NCARSCLOSDTOP=2..3] [NCARSOPENREC=0..1] [NCARSELLIPSE=0] [NCARSSHORT=1..3] [NCARSLONG=2] [NCARSHEXLOD=0] [NCARSCIRLOD=1..2] [NCARSNPL1=2..4] [NCARSWTYPEMXD=0] [NCARSWTYPECL=3..5] [NCARSWTYPEBK=0..1] [DIFFCSHAPE=3..4] [DIFFLSHAPE=2] [DIFFNPL=2] [LSHAPE4<>HEXLOD] [WTYPE4<>BK] [TOXIC=T]	3	-89	0
		EVENTS COVERED: 3,7,8,9			
1	TOTALS			-280	0
4	1	[CSHAPE2=OPENRECT] [CSHAPE3=SLOPETOP] [FANWHL2=0] [FAWTYPEBK=1] [FAWTYPECL=0] [FALSHAPERECT=0] [LN2=LONG] [LN3=SHORT] [LSHAPE2=RECTLOD] [LSHAPE3=TRILOD] [NCAR=5] [NPL2=3] [NPL3=1] [NWHL2=2] [SAMENWHL123=1] [SAMEWTYPE123=1] [WTYPE3=BK] [WTYPE1=BK] [WTYPE2=BK] [NCARSCLOSDREC=0] [NCARSJAGGED=0] [NCARSSLOPE=1] [NCARSOPENTOP=3] [NCARSCLOSDTOP=2] [NCARSOPENREC=3] [NCARSOPENTRAP=0] [NCARSUSHAPE=0] [NCARSDBLOPREC=0] [NCARSHEXAGON=0] [NCARSELLIPSE=0] [NCARSSHORT=2] [NCARSLONG=3] [NCARSHEXLOD=1] [NCARSCIRLOD=1] [NCARSRECTLOD=1] [NCARSTRILOD=1] [NCARSNPL2=0] [NCARSNPL3=1] [NCARSNPLO=1] [NCARSNPL1=3] [NCARSNWHL2=4] [NCARSNWHL3=1] [NCARSWTYPEMXD=0] [NCARSWTYPECL=0] [NCARSWTYPEBK=5] [DIFFCSHAPE=3] [DIFFLSHAPE=4] [DIFFNPL=3] [DIFFNWHL=2] [DIFFWTYPE=1]	1	-100	-1

```
        [LSTINFRONT3=0] [NEXTRA1CAR=0]
        [NEXTRA2CAR=1] [CSHAPE4=OPENRECT]
        [LSHAPE4=HEXLOD] [NPL4=1] [NWHL4=3]
        [WTYPE4=BK] [TOXIC=T]
        EVENTS COVERED: 1
 4   2  [CSHAPE2<>OPENRECT]                          4          8        -1
        [CSHAPE3<>SLOPETOP] [FAWTYPEBK=0]
        [FAWTYPECL=1] [NPL2=1] [SAMEWTYPE123=1]
        [WTYPE3=CL] [WTYPE1=CL] [WTYPE2=CL]
        [NCARSCLOSDREC=0  1] [NCARSSLOPE=0]
        [NCARSCLOSDTOP=1..2]
        [NCARSOPENREC=0  1] [NCARSHEXAGON=0]
        [NCARSLONG=1  2] [NCARSHEXLOD=0]
        [NCARSRECTLOD=1..2] [NCARSNPL2=0  1]
        [NCARSNPL3=0] [NCARSNPLO=1]
        [NCARSWTYPEMXD=0] [NCARSWTYPECL=3..5]
        [NCARSWTYPEBK=0] [DIFFLSHAPE=1..2]
        [DIFFWTYPE=1] [NEXTRA1CAR=0]
        [LSHAPE4<>HEXLOD] [NPL4<>0] [NWHL4<>3]
        [WTYPE4<>BK]
        EVENTS COVERED: 4,8,9,10
 4   3  [CSHAPE3<>SLOPETOP] [FAWTYPEBK=0]           7          17        -1
        [FAWTYPECL=0] [FALSHAPERECT=0]
        [LSHAPE2=TRILOD,CIRLOD]
        [LSHAPE3=TRILOD,RECTLOD] [NCAR=3  4]
        [NPL3=1] [NWHL2=2] [SAMEWTYPE123=0]
        [NCARSSLOPE=0] [NCARSOPENTOP=1..2]
        [NCARSCLOSDTOP=2..3]
        [NCARSOPENREC=0..1]
        [NCARSOPENTRAP=0..1] [NCARSELLIPSE=0]
        [NCARSSHORT=1..3] [NCARSLONG=1..2]
        [NCARSHEXLOD=0] [NCARSCIRLOD=1]
        [NCARSRECTLOD=0..1] [NCARSTRILOD=1..2]
        [NCARSNPL1=1..3] [NCARSNWHL2=3..4]
        [NCARSWTYPECL=0..3]
        [NCARSWTYPEBK=1..3] [DIFFCSHAPE=3..4]
        [DIFFLSHAPE=2..3] [DIFFWTYPE=2]
        [NEXTRA2CAR=0] [CSHAPE4<>OPENRECT]
        [LSHAPE4<>HEXLOD]
        EVENTS COVERED: 2,3,5,6,7
 4   TOTALS                                                -75        -3

(68 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  1.7E+022

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPERIMENT 1: K=4, CRITERION=DS

THE 2 BEST CLUSTERINGS FOLLOW...    (319009 MS)
```

ITER	CPLX#	VL-RULE	SEED	COSTS SIM	COSTS DIM
1	1	[CSHAPE2=OPENRECT] [CSHAPE3=SLOPETOP] [FANWHL2=0] [FAWTYPEBK=1] [FAWTYPECL=0] [FALSHAPERECT=0] [LN2=LONG] [LN3=SHORT] [LSHAPE2=RECTLOD] [LSHAPE3=TRILOD] [NCAR=5] [NPL2=3] [NPL3=1] [NWHL2=2] [SAMENWHL123=1] [SAMEWTYPE123=1] [WTYPE3=BK] [WTYPE1=BK] [WTYPE2=BK] [NCARSCLOSDREC=0] [NCARSJAGGED=0] [NCARSSLOPE=1] [NCARSOPENTOP=3] [NCARSCLOSDTOP=2] [NCARSOPENREC=3] [NCARSOPENTRAP=0] [NCARSUSHAPE=0] [NCARSDBLOPREC=0] [NCARSHEXAGON=0] [NCARSELLIPSE=0] [NCARSSHORT=2] [NCARSLONG=3] [NCARSHEXLOD=1] [NCARSCIRLOD=1] [NCARSRECTLOD=1] [NCARSTRILOD=1] [NCARSNPL2=0] [NCARSNPL3=1] [NCARSNPL0=1] [NCARSNPL1=3] [NCARSNWHL2=4] [NCARSNWHL3=1] [NCARSWTYPEMXD=0] [NCARSWTYPECL=0] [NCARSWTYPEBK=5] [DIFFCSHAPE=3] [DIFFLSHAPE=4] [DIFFNPL=3] [DIFFNWHL=2] [DIFFWTYPE=1] [LSTINFRONT3=0] [NEXTRA1CAR=0] [NEXTRA2CAR=1] [CSHAPE4=OPENRECT] [LSHAPE4=HEXLOD] [NPL4=1] [NWHL4=3] [WTYPE4=BK] [TOXIC=T] EVENTS COVERED: 1	1	-100	0
1	2	[CSHAPE2=OPENTOP] [CSHAPE3<>SLOPETOP] [FAWTYPEBK=0] [FAWTYPECL=0] [FALSHAPERECT=0] [LN2=SHORT] [LSHAPE2=TRILOD] [LSHAPE3=RECTLOD] [NCAR=4] [NPL2=1] [NPL3=1] [NWHL2=2] [SAMEWTYPE123=0] [WTYPE1=BK,MXD] [NCARSCLOSDREC=1..2] [NCARSJAGGED=0] [NCARSSLOPE=0] [NCARSOPENTOP=1..2] [NCARSCLOSDTOP=2..3] [NCARSOPENREC=0] [NCARSOPENTRAP=0..1] [NCARSHEXAGON=0] [NCARSELLIPSE=0] [NCARSSHORT=2..3] [NCARSLONG=1..2] [NCARSHEXLOD=0] [NCARSCIRLOD=1] [NCARSRECTLOD=1] [NCARSTRILOD=1] [NCARSNPL2=0..1] [NCARSNPL0=1] [NCARSNPL1=2..3] [NCARSNWHL2=3..4] [NCARSWTYPECL=0..3] [NCARSWTYPEBK=1..3] [DIFFCSHAPE=3..4] [DIFFLSHAPE=3] [DIFFWTYPE=2] [LSTINFRONT3=0] [NEXTRA1CAR=1] [NEXTRA2CAR=0] [CSHAPE4=CLOSDRECT] [LSHAPE4=CIRLOD] [NPL4=EXISTS] [NWHL4=2] [WTYPE4=EXISTS] [TOXIC=F] EVENTS COVERED: 2,5	2	-87	0

1	3	[CSHAPE3<>SLOPETOP] [FAWTYPEBK=0] [FALSHAPERECT=0] [LSHAPE2=RECTLOD, CIRLOD] [NPL2=1] [NPL3=1] [WTYPE3=CL] [WTYPE1=BK,CL] [NCARSCLOSDREC=0 .1] [NCARSSLOPE=0] [NCARSCLOSDTOP=2 .3] [NCARSOPENREC=0. 1] [NCARSELLIPSE=0] [NCARSSHORT=1 .3] [NCARSLONG=2] [NCARSHEXLOD=0] [NCARSCIRLOD=1..2] [NCARSNPL1=2 .4] [NCARSWTYPEMXD=0] [NCARSWTYPECL=3. 5] [NCARSWTYPEBK=0..1] [DIFFCSHAPE=3..4] [DIFFLSHAPE=2] [DIFFNPL=2] [LSHAPE4<>HEXLOD] [WTYPE4<>BK] [TOXIC=T] EVENTS COVERED: 3,7,8,9	3	-89	0
1	4	[CSHAPE2<>OPENRECT] [CSHAPE3=OPENTOP] [FANWHL2=1] [FAWTYPEBK=0] [LSHAPE3=TRILOD,RECTLOD] [NWHL2=2] [SAMENWHL123=1] [WTYPE3=CL] [WTYPE1=BK, CL] [WTYPE2=CL] [NCARSCLOSDREC=0..1] [NCARSJAGGED=0] [NCARSSLOPE=0] [NCARSCLOSDTOP=1..2] [NCARSOPENREC=1] [NCARSOPENTRAP=0..1] [NCARSHEXAGON=0] [NCARSLONG=1 .2] [NCARSHEXLOD=0] [NCARSCIRLOD=0..1] [NCARSNPL2=0..1] [NCARSNPL0=1] [NCARSNWHL2=3..5] [NCARSNWHL3=0] [NCARSWTYPEMXD=0] [NCARSWTYPECL=2..5] [NCARSWTYPEBK=0. 1] [DIFFLSHAPE=1..2] [DIFFNWHL=1] [NEXTRA1CAR=0] [CSHAPE4<>OPENRECT] [LSHAPE4<>HEXLOD] [NPL4<>0] [NWHL4<>3] [WTYPE4<>BK] [TOXIC=F] EVENTS COVERED: 4,6,10	4	-88	0
1	TOTALS			-364	0
3	1	[CSHAPE2=OPENRECT] [CSHAPE3=SLOPETOP] [FANWHL2=0] [FAWTYPEBK=1] [FAWTYPECL=0] [FALSHAPERECT=0] [LN2=LONG] [LN3=SHORT] [LSHAPE2=RECTLOD] [LSHAPE3=TRILOD] [NCAR=5] [NPL2=3] [NPL3=1] [NWHL2=2] [SAMENWHL123=1] [SAMEWTYPE123=1] [WTYPE3=BK] [WTYPE1=BK] [WTYPE2=BK] [NCARSCLOSDREC=0] [NCARSJAGGED=0] [NCARSSLOPE=1] [NCARSOPENTOP=3] [NCARSCLOSDTOP=2] [NCARSOPENREC=3] [NCARSOPENTRAP=0] [NCARSUSHAPE=0] [NCARSDBLOPREC=0] [NCARSHEXAGON=0] [NCARSELLIPSE=0] [NCARSSHORT=2] [NCARSLONG=3] [NCARSHEXLOD=1] [NCARSCIRLOD=1] [NCARSRECTLOD=1] [NCARSTRILOD=1] [NCARSNPL2=0]	1	-100	0

[NCARSNPL3=1] [NCARSNPLO=1]
[NCARSNPL1=3] [NCARSNWHL2=4]
[NCARSNWHL3=1] [NCARSWTYPEMXD=0]
[NCARSWTYPECL=0] [NCARSWTYPEBK=5]
[DIFFCSHAPE=3] [DIFFLSHAPE=4]
[DIFFNPL=3] [DIFFNWHL=2] [DIFFWTYPE=1]
[LSTINFRONT3=0] [NEXTRA1CAR=0]
[NEXTRA2CAR=1] [CSHAPE4=OPENRECT]
[LSHAPE4=HEXLOD] [NPL4=1] [NWHL4=3]
[WTYPE4=BK] [TOXIC=T]
EVENTS COVERED: 1

3 2 [CSHAPE2<>OPENRECT] [CSHAPE3=OPENTOP] 2 -85 0
[FANWHL2=1] [FAWTYPEBK=0] [FAWTYPECL=0]
[FALSHAPERECT=0] [LN3=SHORT]
[LSHAPE2=TRILOD,CIRLOD]
[LSHAPE3=TRILOD,RECTLOD] [NCAR=3 4]
[NPL3=1] [NWHL2=2] [SAMENWHL123=1]
[SAMEWTYPE123=0] [WTYPE3=CL]
[WTYPE1=BK] [WTYPE2=CL]
[NCARSCLOSDREC=1] [NCARSJAGGED=0]
[NCARSSLOPE=0] [NCARSOPENTOP=1..2]
[NCARSCLOSDTOP=2] [NCARSOPENREC=0..1]
[NCARSOPENTRAP=0..1] [NCARSDBLOPREC=0]
[NCARSHEXAGON=0] [NCARSELLIPSE=0]
[NCARSSHORT=1..3] [NCARSLONG=1..2]
[NCARSHEXLOD=0] [NCARSCIRLOD=1]
[NCARSRECTLOD=0..1] [NCARSTRILOD=1]
[NCARSNPL2=0..1] [NCARSNPLO=1]
[NCARSNPL1=1..2] [NCARSNWHL2=3..4]
[NCARSNWHL3=0] [NCARSWTYPEMXD=0]
[NCARSWTYPECL=2..3] [NCARSWTYPEBK=1]
[DIFFCSHAPE=3 4] [DIFFLSHAPE=2..3]
[DIFFNPL=3] [DIFFNWHL=1] [DIFFWTYPE=2]
[NEXTRA2CAR=0] [CSHAPE4<>OPENRECT]
[LSHAPE4<>HEXLOD] [NPL4<>0] [NWHL4<>3]
[WTYPE4<>BK] [TOXIC=F]
EVENTS COVERED: 2,6

3 3 [CSHAPE3<>SLOPETOP] [FAWTYPEBK=0] 9 -89 0
[FALSHAPERECT=0] [LSHAPE2=RECTLOD,
CIRLOD] [NPL2=1] [NPL3=1] [WTYPE3=CL]
[WTYPE1=BK,CL] [NCARSCLOSDREC=0 1]
[NCARSSLOPE=0] [NCARSCLOSDTOP=2..3]
[NCARSOPENREC=0..1] [NCARSELLIPSE=0]
[NCARSSHORT=1..3] [NCARSLONG=2]
[NCARSHEXLOD=0] [NCARSCIRLOD=1..2]
[NCARSNPL1=2..4] [NCARSWTYPEMXD=0]
[NCARSWTYPECL=3..5]
[NCARSWTYPEBK=0..1] [DIFFCSHAPE=3..4]
[DIFFLSHAPE=2] [DIFFNPL=2]
[LSHAPE4<>HEXLOD] [WTYPE4<>BK]
[TOXIC=T]
EVENTS COVERED: 3,7,8,9

```
 3    4  [CSHAPE2=OPENTOP][CSHAPE3<>SLOPETOP]      4          -88          0
         [FAWTYPEBK=0][LN2=SHORT]
         [LSHAPE2=RECTLOD,TRILOD]
         [LSHAPE3=TRILOD,RECTLOD][NPL2=1]
         [NWHL2=2][WTYPE1=CL,MXD]
         [NCARSJAGGED=0][NCARSSLOPE=0]
         [NCARSOPENREC=0, 1]
         [NCARSOPENTRAP=0, 1][NCARSHEXAGON=0]
         [NCARSLONG=1, 2][NCARSHEXLOD=0]
         [NCARSCIRLOD=0, 1][NCARSRECTLOD=1, 2]
         [NCARSNPL2=0, 1][NCARSNPLO=1]
         [NCARSNWHL2=3, 5][NCARSWTYPEBK=0, 3]
         [DIFFLSHAPE=1, 3][CSHAPE4<>OPENRECT]
         [NPL4<>0][NWHL4<>3][TOXIC=F]
         EVENTS COVERED. 4,5,10
 3   TOTALS                                                -362          0

(182 STARS BUILT)

FOR THE BEST SOLUTION ABOVE, S=  9.2E+019

ACTIVITY STATISTICS
CENDIST   =    18  CRITVAL   = 29992  CLUSTER   =    15
CMPCOV    =     0  EXTEND    = 13931  FREECPLX  = 16261  GENRLIZ   = 20966
INTRSCT   = 19803  NID       =   359  NUMSEL    =     0  NEWCPLX   = 16751
REDUCE    = 13931  REFHIGH   =217266  REFNUM    =1769528  REFUNION  = 69419
REFLEV    =2298741  REFLOW    =242492  SETMAP    = 29992  SETLEVELMA=     0
STAR      =   266  SEMANTICS =  1548  SYNDIST   =     0  TRIM      =   698
TCOVER    =439230  DEGISCT   =    0  CLUSTERING=     1  TABLESETUP=    51
BESTC     =  7367  MCARD     = 43217  GENPATH   =   376  CLEARCV   =     6
SAVECV    =    15  STAR2     =     0
```

REFERENCES

[Anderberg, 1973]
Anderberg, M.R., *Cluster Analysis for Appliations*, New York: Academic, 1973.

[Barsalou, 1982]
Barsalou, L.W., "Goal-directed information abstraction during conceptual organization," Symp. on Psych. Processes in Categorization, Conf. of the Society for Math. Psych., princeton University, August, 1982.

[Buchanan et al., 1976]
Buchanan, B.G., Smith, D.H., White, W.C., Griter, R.J., Geigenbaum, E.A., Lederberg, J., Djerassi, C., "Automatic rule formation in mass spectrometry by means of the Meta-Dentral program," *Journal of the American Chemical Society*, Vol. 98, pp. 6168, 1976.

[Buchanan and Mitchell, 1978]
Buchanan, G.B., Mitchell, T.M., "Model-directed learning of production rules," in *Pattern-Directed Inference Systems*, D.A. Waterman and F. Hayes-Roth (Eds.), New York: Academic, pp. 297-312, 1978.

[Carbonell, 1983]
Carbonell, J.G., "Learning by analogy: Formulating and generalizing plans from past experience," in *MACHINE LEARNING: An Artificial Intelligence Approach*, R. S. Michalski, J. Carbonell and T. Mitchell (Eds.), Palo Alto, Calif.: Tioga, 1983.

[Cohen, 1977]
Cohen, B.L., "A powerful and efficient structural pattern recognition system," *Artificial Intelligence*, vol. 9, December, pp. 223-255, 1977.

[Davis and Lenat, 1982]
Davis, R., Lenat, D.B., *Knowledge-based Systems in Artificial Intelligence*, New York: McGraw-Hill, 1982.

[Diday and Simon, 1976]
Diday, E., Simon, J.C., "Clustering analysis," *Communication and Cybernetics 10*, New York: Springer-Verlag, 1976

[Diday, 1978]
Diday, E., "Problems of clustering and recent advances," 11th Congress of Statistics, Oslo, Norway, 1978.

[Dietterich and Michalski, 1981]
Dietterich, T.G., Michalski, R.S., "Inductive learning of structural descriptions: Evaluation criteria and comparative review of selected methods," *Artificial Intelligence*, vol. 16, July, pp. 257-294, 1981.

[Doran and Michie, 1966]
Doran, J., Michie, D., "Experiments with the graph-traverser program," Proceedings of the Royal Society, pp. 235-259, 1966.

[Duda and Hart, 1973]
Duda, R., Hart. P.E., *Pattern Classificationa and Scene Analysis*, John Wiley, 1973.

[Gowda and Krishna, 1978]
Gowda, K.C., Krishna, G., "Disaggregative clustering using the concept of mutual nearest neighborhood," *Man and Cybernetics*, IEEE Transactions on Systems, pp. 888-894, December 1978.

[Hayes-Roth, 1976]
Hayes-Roth, F., "Patterns of induction and associated knowledge acquisition algorithms," Technical Report, Carnegie-Mellon University, May 1976.

[Hayes-Roth and McDermott, 1977]
Hayes-Roth, F., McDermott, J., "Knowledge acquisition from structural descriptions," *Fifth Int. Joint Conf. Artificial Intelligence*, pp. 356-362, 1977.

[Hoff, Michalski and Stepp, 1983]
Hoff, W., Michalski, R.S., Stepp, R., "INDUCE 2: A Program for Learning Structural Descriptions from Examples," Urbana Illinois: Univ. of Illinois Dept. of Computer Science Tech. Rept. No. UIUCDCS-F-83-904, January, 1983.

[Langley, Bradshaw and Simon, 1983]
Langley, P., Bradshaw, G.L., Simon, H.A., "Rediscovering Chemistry With the BACON System," in *MACHINE LEARNING: An Artificial Intelligence Approach*, R. S. Michalski, J. Carbonell and T. Mitchell (Eds.), Palo Alto, Calif.: Tioga, 1983.

[Larson, 1977]
Larson, J.B., "Inductive Inference in the Variable-Valued Predicate Logic System VL_{21}: Methodology and Computer Implementation," Ph.D. Thesis, Report No. 869, Department of Computer Science, University of Illinois, Urbana, Illinois, May 1977.

[Larson and Michalski, 1977]
Larson, J.B., Michalski, R.S., "Inductive Inference of VL Decision Rules," Invited paper for the *Workshop in Pattern-Directed Inference Systems, Hawaii, May 23-27, 1977* and *published in* SIGART Newsletter, ACM, No. 63, pp. 38-44, June 1977.

[Lenat, 1983]
Lenat, D.B., "The Role of Heuristics in Learning by Discovery: Three Case Studies," in *MACHINE LEARNING: An Artificial Intelligence Approach*, R. S. Michalski, J. Carbonell and T. Mitchell (Eds.), Palo Alto, Calif.: Tioga, 1983.

[MacQueen, 1967]
MacQueen, J., "Some methods for classification and analysis on multivariate observations," *Proc. of the 5th Berkley Symp. on Statistics and Probability*, Berkley Calif.: Univ. of Calif. Press, 1967.

[Medin, Wattenmaker and Michalski, 1984]
Medin, D.L., Wattenmaker, W.S., Michalski, R.S., "Constraints on Rule Induction in Classification," unpublished manuscript, 1984.

[Meisel, 1972]
Meisel, W., *Computer oriented approaches to pattern recognition*, New York: Academic Press, 1972.

[Michalski, 1972]
Michalski, R.S., "A Variable-Valued Logic System as Applied to Picture Description and Recognition," Chapter in the book, *Graphic Languages*, F. Nake and A. Rosenfeld (Eds.), North-Holland Publishing Co., 1972.

[Michalski, 1974]
Michalski, R.S., "VARIABLE-VALUED LOGIC: System VL_1," *Proceedings of the 1974 International Symposium on Multiple-Valued Logic*, West Virginia University, Morgantown, West Virginia, May 29-31, 1974.

[Michalski, 1975]
Michalski, R.S., "Variable-Valued Logic and Its Applications to Pattern Recognition and Machine Learning," Chapter in the monograph, *Computer Science and Multiple-Valued Logic Theory and Applications*, D. C. Rine (Ed.), North-Holland Publishing Co., 1975.

[Michalski, 1980a]
Michalski, R.S., "Pattern recognition as rule-quided inductive inference," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-2, July, pp. 349-361, 1980.

[Michalski, 1980b]
Michalski, R.S., "Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts," *J. Policy Analysis and Information Systems*, vol. 4., Sept., pp. 219-244, 1980.

[Michalski, 1983]
Michalski, R.S., "A theory and methodology of inductive learning," in *MACHINE LEARNING: An Artificial Intelligence Aproach*, R.S. Michalski, J. Carbonell and T. Mitchell (Eds.), Palo Alto, Calif.: Tioga, 1983.

[Michalski, Baskin and Spackman, 1982]
Michalski, R.S., Baskin, A.B., Spackman, K.A., "A Logic-based Approach to Conceptual Database Analysis." *Sixth Annual Symposium on Computer Applications in Medical Care (SCAMC-6)*, George Washington Univ. Medical Center, Washington, DC, November 1-2, pp. 792-796, 1982.

[Michalski and Stepp, 1983]
Michalski, R.S., Stepp, R.E., "LEARNING FROM OBSERVATION: Conceptual Clustering," in *MACHINE LEARNING: An Artificial Intelligence Approach*, R. S. Michalski, J. Carbonell and T. Mitchell (Eds.), TIOGA Publishing Co., Palo Alto, 1983.

[Michalski, Stepp and Diday, 1981]
Michalski, R.S., Stepp, R.E., Diday, E., "A recent advance in data analysis: Clustering objects into classes characterized by conjunctive concepts," in *Progress in Pattern Recognition*, vol. 1, L.N. Kanal and A. Rosenfeld (Eds.), New York: North-Holland, pp. 33-56, 1981.

[Nilsson, 1980]
Nilsson, N.J., *Principles of Artificial Intelligence*, Palo Alto, Calif.: Tioga, 1980.

[Paterson, 1983]
Paterson, A., "An Application of CLUSTER and GEM Learning Programs to the Synthesis of Decision Structures for the KPK Chess Endgame," Urbana, Illinois: Univ. of Illinois Dept. of Computer Science Tech. Rept. No. UIUCDCS-R-83-1156, December, 1983.

[Rendell, 1983]
Rendell, L.A., "Toward a unified approach for conceptual knowledge acquisition," The AI Magazine, Winter 1983.

[Sokal and Sneath, 1963]
Sokal, R.R., Sneath, P.H., *Principles of Numerical Taxonomy*, San Francisco: W.H. Freeman, 1963.

[Stepp, 1979]
Stepp, R., "The Uniclass Inductive Program AQ7UNI: Program Implementation and User's Guide," Report No. 949, Department of Computer Science, University of Illinois, Urbana, IL, July 1979.

[Stepp, 1980]
Stepp, R., "Learning from Observations: Experiments in Conceptual Clustering," *Workshop on Current Developments in Machine Learning*, Carnegie-Mellon University, Pittsburgh, July 16-18, 1980.

[Stepp, 1984]
Stepp, R., "A Description and User's Guide for CLUSTER/2, A Program for Conjunctive Conceptual Clustering," Urbana Illinois: Univerisity of Illinois Dept. of Computer Science Technical Report No. UIUCDCS-R-84-1084, 1984.

[Utgoff, 1984]
Utgoff, P.E., "Shift of Bias for Inductive Concept Learning," *Machine Learning, Book II*, R.S. Michalski, J. Carbonnel, T. Mitchell (Eds.), Tioga Publishing Company, 1984.

[Vere, 1975]
Vere, S.A., "Induction of concepts in the predicate calculus," *Proceedings of the Fourth International Conf. on Artificial* Intelligence, IJCAI, Tbilisi, USSR, 1975.

[Winston, 1975]
Winston, P.H., "Learning structural descriptions from examples," Chapter in the book *The psychology of Computer Vision*, P.H. Winston (Ed.), McGraw Hill, 1975.

[Winston, 1977]
Winston, P.H., *Artificial Intelligence*, Addison-Wesley, 1977.

[Winston, 1979]
Winston, P.H., "Learning and reasoning by analogy," *Communications of the ACM*, Vol. 23, No. 12, pp. 689-703, 1979.

[Zadeh, 1965]
Zadeh, L.A., "Fuzzy Sets," *Information and Control*, Vol. 8, No. 3, 1965.

VITA

Robert Earl Stepp, III was born on April 15, 1948 in Lincoln Nebraska. He grew up there and attended the University of Nebraska—Lincoln. In 1970 he received the A.B. degree with a major in Physics and in 1971 he received the M.S. degree with a major in Computer Science. Beginning in January, 1972 he worked for the University of Nebraska Computer Network (a multi-campus computing facility) and later that year became the manager of systems programming. In 1976 Dr. Stepp moved to Champaign Illinois to resume graduate study in Computer Science. He received the Ph.D. degree from the University of Illinois at Urbana-Champaign in 1984.

Dr. Stepp's interests include machine acquisition and representation of knowledge, conceptual data analysis, man-computer interaction, software engineering, computing to aid the handicapped, personal computing, music, and electronics. He is also a part-time professional symphony musician who plays the double bass.

Dr. Stepp presently holds the positions of Assistant Professor of Electrical and Computer Engineering and Research Assistant Professor in the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign.

END

DATE

FILMED

DEC.

1987